INVESTIGATIONS ON EFFICIENT ADAPTATION ALGORITHMS

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Murat Belge

September 1995

INVESTIGATIONS ON EFFICIENT ADAPTATION ALGORITHMS

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND

ELECTRONICS ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Murat Belge

September 1995

ABSTRACT

INVESTIGATIONS ON EFFICIENT ADAPTATION ALGORITHMS

Murat Belge
M.S. in Electrical and Electronics Engineering
Supervisor: Assist. Prof. Dr. Orhan Arıkan
September 1995

Efficient adaptation algorithms, which are intended to improve the performances of the LMS and the RLS algorithms are introduced.

It is shown that nonlinear transformations of the input and the desired signals by a softlimiter improve the convergence speed of the LMS algorithm at no cost, with a small bias in the optimal filter coefficients. Also, the new algorithm can be used to filter α-stable non-Gaussian processes for which the conventional adaptive algorithms are useless.

In a second approach, a prewhitening filter is used to increase the convergence speed of the LMS algorithm. It is shown that prewhitening does not change the relation between the input and the desired signals provided that the relation is a linear one. A low order adaptive prewhitening filter can provide significant speed up in the convergence.

Finally, adaptive filtering algorithms running on roughly quantized signals are proposed to decrease the number of multiplications in the LMS and the RLS algorithms. Although, they require significantly less computations their preformances are comparable to those of the conventional LMS and RLS algorithms.

ÖZET

VERİMLİ UYARLAMA ALGORİTMALARININ ARAŞTIRILMASI

Murat Belge
Elektrik ve Elektronik Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Dr. Orhan Arıkan
Eylül 1995

EOK (LMS) ve ÖEK (RLS) uyarlama algoritmalarının performanslarını geliştirmek amacıyla, verimli uyarlama algoritmaları tanıtılmıştır.

Giriş ve referans işaretlerinin doğrusal olmayan bir dönüşüm yolu ile yumuşak sınırlandırıcıdan geçirilmesinin, işlem karmaşıklığını arttırmadan en iyi süzgeç parametrelerinde küçük bir sapma ile, EOK algoritmasının yakınsama hızını arttırdığı gösterilmiştir. Yeni tanımlanan algoritma, birçok uyarlama algoritmasının çalışmadığı Gauss dağılımına sahip olmayan rastgele süreçlerin süzgeçlenmesinde kullanılabilir.

İkinci bir yaklaşımda, bir ön beyazlaştırıcı filtre EOK algoritmasının yakınsama hızının arttırılması amacıyla kullanılmıştır. Ön beyazlaştırıcı filtrenin giriş ve referans işaretleri arasındaki lineer bir ilişkiyi değiştirmediği gösterilmiştir. Düşük dereceli bir ön beyazlaştırıcı filtre EOK algoritmasının yakınsama hızını önemli bir derecede arttırabilmektedir.

Sonuncu olarak, EOK ve ÖEK algoritmalarının hesap karmaşıklığının azaltılması için, kabaca seviyelendirilmiş işaretler üzerinde çalışan uyarlamalı süzgeç algoritmaları onerilmiştir. Bu algoritmalar, klasik EOK ve ÖEK algoritmalarından çok daha az hesap gerektirmelerine ragmen performansları bu algoritmaların performansları ile kıyaslanabilir derecededir.

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Orhan Arıkan(Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Enis Çetin

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Billur Barshan

Approved for the Institute of Engineering and Sciences:

Prof. Dr. Mehmet Baray
Director of Institute of Engineering and Sciences

ACKNOWLEDGEMENT

I would like to express my deep gratitude to my supervisor Dr. Orhan Arıkan and Dr. Enis Çetin and Dr. Billur Barshan for their guidance, suggestions, and invaluable encouragement throughout the development of this thesis.

And special thanks to all Graduate Students in this department for their continuous support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

The term *filtering* is used to describe an operation which is applied to a set of noisy data in order to extract information about a prescribed quantity of interest. The data may come from, for example, a noisy communication channel or from noisy sensors. A filter is characterized by a set of parameters adjusted to produce a desired response. If the output of the filter is a linear function of its input, it is said to be linear.

In the statistical approach to the linear filtering problem, we assume the availability of some statistical parameters of input data (mean, correlation, etc.) and derive the filter that would yield the best performance according to some statistical optimality criteria. A useful approach to the linear filter optimization problem is to minimize the expected value of the error squared defined as the difference between some desired response and the output of the filter. In a stationary environment, the resulting solution is commonly known as the Wiener filter[2].

The design of Wiener filter requires a priori information about the statistics of the input data. If the statistical parameters used to derive the filter deviate from the actual values, Wiener filter is no longer optimal. In such situations, one has to estimate the underlying statistics of the input data and plug in the filter to ensure proper operation. However such a direct approach is costly since it needs elaborate hardware and computation. A second approach is to use an *adaptive filter*. An adaptive filter, as the name implies, is a learning and self designing system which adjust its parameters by a recursive algorithm to optimize some performance criteria. They are learning systems in the sense

that they sense and learn their unique operation environment and adjust their parameters so that the system will be optimized.

The first adaptive filtering algorithm, known as the Least Mean Square(LMS) algorithm, was introduced by Widrow and Hoff [36]. It is a simple, easily understandable and robust algorithm. Even though it might converge slowly when the input signal is highly correlated, this simple algorithm contributed much to the development and application of digital adaptive algorithms.

Another major contribution to the development of adaptive filtering algorithms was made by Goddart in 1974[2]. He used Kalman filter theory to propose a new class of adaptive filtering algorithms for obtaining rapid convergence of tap weights of a transversal filter to their optimum settings. The Kalman algorithm is closely related to the recursive least squares(RLS) algorithm that follows from the method of least squares. RLS algorithm usually provides much faster rate of convergence than the LMS algorithm at the expense of increased computational complexity.

In this thesis we investigate some of the topics in adaptive signal processing. Each topic is presented as a chapter which has its own introduction and conclusion.

In the second Chapter, we propose a new adaptive algorithm running on nonlinearly transformed input and desired signals based on the conventional LMS algorithm. We show that the convergence properties of LMS algorithm can be improved substantially. The proposed configuration performs better than the conventional LMS in situations where the input signals are corrupted by additive impulsive interference. The proposed method can also be used to filter non-Gaussian α-stable processes.

In Chapter 3, we investigate a new adaptive configuration, which consists of a prewhitening filter followed by an adaptive section which uses LMS as the adaptation algorithm. It is shown that prewhitening filter acts as a decorrelator and reduces the condition number of the input autocorrelation matrix hence increases the convergence speed of overall adaptation. A small order adaptive prewhitening filter is sufficient for an impressive speed up in the convergence.

Finally, in the fourth Chapter we investigate adaptive filtering algorithms, which use roughly quantized signals (4,3 even 1-bit quantized signals) as their inputs. Theoretically, it is shown that these algorithms solve exactly the same normal equations corresponding to the RLS case. In these new algorithms, the

number of multiplications required to update adaptive filter parameters are significantly reduced without degrading the performance.

Chapter 2

INVESTIGATION OF NONLINEAR PRE-FILTERING IN LEAST MEAN SQUARE ADAPTATION

2.1 Introduction

The LMS algorithm first introduced by Widrow and Hoff in 1959 [35] has been one of the most popular algorithms for adaptive filtering because of its conceptual and computational simplicity and robustness. It does not require measurement of pertinent correlation functions, nor does it require matrix inversion. Unfortunately, the convergence rate is highly dependent on the conditioning of the autocorrelation matrix of filter inputs. The mean square error of the adaptive filter trained with LMS decreases over time as a sum of exponentials whose time constants are inversely proportional to the eigenvalues of the autocorrelation matrix of the filter inputs. Thus, small eigenvalues create slow

convergence modes. On the other hand the largest eigenvalue puts an upper bound on the parameters governing the learning rate without encountering instability problems. It results from these two counteracting forces that the best convergence properties are obtained when all of the eigenvalues of the input autocorrelation matrix are equal, in which case input is called white. As the eigenvalue spread of the input autocorrelation matrix increases, the convergence speed of the LMS algorithm deteriorates.

In the past, there have been attempts to overcome slow convergence problem of the LMS algorithm by applying preprocessing techniques which were expected to decorrelate the input signals in time. Transform-domain adaptive filters which transform inputs with a linear orthogonal transform such as DFT or DCT have been introduced in this context[41-42-43]. The performance of these algorithms depend on the orthogonalizing capability of the data-independent transformation. Although they can provide significant improvement in the convergence speed, they introduce additional computational requirements.

In this chapter we will introduce a novel adaptation algorithm which is essentially a nonlinear transformation followed by the LMS adaptation. Through numerous simulations, it has been observed that the conditioning of the input autocorrelation matrix could be ameliorated. The new configuration is found to be useful in case of impulsive interference and filtering α-stable processes.

2.2 The LMS Algorithm

A typical adaptive filtering configuration is shown in Fig. (2.1). The conventional LMS algorithm has the following simple update:

$$\underline{w}(n+1) = \underline{w}(n) + \mu e(n)\underline{x}(n) \quad , \tag{2.1}$$

where μ is an adaptation constant, $\underline{w}(n)$ is the column vector of the tap weights of the adaptive filter at time n and $\underline{x}(n)$ is the column vector of the N most recent samples of the input at time n. i.e.

$$\underline{w}(n) = [w_1(n)\ w_2(n) \ldots w_N(n)]^T \tag{2.2}$$

$$\underline{x}(n) = [x(n)\ x(n-1)\ \ldots x(n-N+1)]^T \tag{2.3}$$

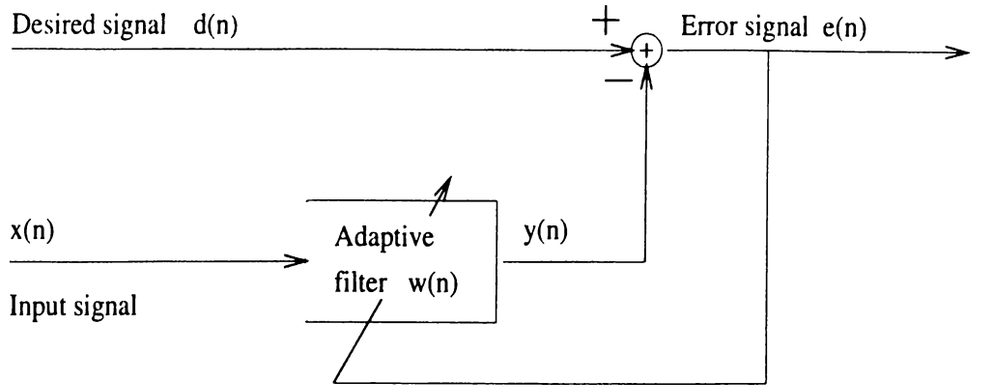

Figure 2.1: *Adaptive filtering block diagram*

The output error at time n is given by

$$\epsilon(n) = d(n) - \underline{w}^T(n)\underline{x}(n) \quad . \tag{2.4}$$

Equation (2.1) simply says that the coefficients of the adaptive filter at each iteration equal to the coefficients of the filter at previous iteration plus a term proportional to the product of the error and the data vector. It requires a total of $2N$ multiplications and $2N$ additions to update filter coefficients at each iteration.

In spite of the appearant simplicity of the LMS algorithm, analysis of the behavior of the filter is difficult. Although a considerable amount of work has been done, the results have been obtained for certain limited types of signals. Following results are obtained for white inputs assuming the independence of $\underline{x}(n)$ and $\underline{w}(n)$.

2.2.1 Convergence In the Mean

It has been shown that [2-7] for stationary inputs the LMS algorithm converges to the following optimal Wiener solution provided that it is stable

$$\lim_{n\to\infty} E[\underline{w}(n)] = \underline{w}^* = R_{xx}^{-1}\underline{r}_{xd} \tag{2.5}$$

where $R_{xx} = E[\underline{x}(n)\underline{x}^T(n)]$ and $\underline{r}_{xd} = E[d(n)\underline{x}(n)]$. The stability is guaranteed if the adaptation constant μ satisfies the following condition

$$0 < \mu < \frac{2}{\lambda_{max}} \tag{2.6}$$

where λ_{max} is the maximum eigenvalue of the input auto-correlation matrix R_{xx}. By defining parameter error vector as

$$\underline{v}(n) = E[\underline{w}(n)] - \underline{w}^* \tag{2.7}$$

it was found that the filter coefficients evolve as

$$v_i(n+1) = (1 - \mu\lambda_i)v_i(n); \quad i = 1, 2, \ldots, N \ . \tag{2.8}$$

¿From this equation the time constant for the convergence of the i^{th} mode is estimated to be

$$t_i \approx \frac{1}{\mu\lambda_i} \ . \tag{2.9}$$

The overall speed of the convergence is clearly limited by the slowest converging mode which in turn stems from the smallest eigenvalue:

$$t \approx \frac{1}{\mu\lambda_{min}} \ . \tag{2.10}$$

By substituting the upper bound found for the adaptation constant we get

$$t \approx \frac{\lambda_{max}}{\lambda_{min}} \ . \tag{2.11}$$

This means that the rate of convergence of the LMS algorithm is governed by the eigenvalue spread or the condition number of the input autocorrelation matrix [7].

2.2.2 Convergence In the Mean Square

Although the LMS algorithm converges to the optimal Wiener solution in the steady-state, noise in the adaptation process causes the steady state solution to vary randomly about the minimum point. This results in a steady-state mean square error $J(n)$ which is greater than the minimum mean square error J_{min}. The steady-state mean square error can be expressed in terms of the tap weight error vector as

$$J(n) = J_{min} + (\underline{w}(n) - \underline{w}^*)^T R_{xx} (\underline{w}(n) - \underline{w}^*) \tag{2.12}$$

$$= J_{min} + \underline{v}^T(n) R_{xx} \underline{v}(n) \ . \tag{2.13}$$

We define the *excess mean square error* as the difference between the mean square error, $J(n)$, produced by the adaptive filter at time n and the minimum value that can be achieved J_{min}, which is given by

$$J_{ex} = J(n) - J_{min} \quad . \tag{2.14}$$

The ratio of the excess MSE to the minimum MSE is defined as the misadjustment [2]. It has been shown that for small values of μ the misadjustment is given by

$$M = \mu \mathrm{tr} R_{xx} \quad . \tag{2.15}$$

For larger values of μ, Nehorai and Malah provided the following result [31].

$$M = \frac{\mu \mathrm{tr} R_{xx}}{1 - \mu \mathrm{tr} R_{xx}} \quad . \tag{2.16}$$

It is seen that for $\mu \mathrm{tr} R_{xx} \ll 1$, the misadjustment linearly varies with μ, which is an intuitively evident result.

2.3 The Nonlinear LMS Algorithm (NLMS)

The proposed algorithm is schematically represented in Fig. 2.2, and it has the following weight update equation:

$$\underline{w}(n+1) = \underline{w}(n) + \mu \hat{e}(n) \underline{\hat{x}}(n) \tag{2.17}$$

where

$$\hat{e}(n) = \hat{d}(n) - \underline{\hat{x}}(n)^T \underline{w}(n) \quad , \tag{2.18}$$

and $\underline{\hat{x}}(n)$ and $\hat{d}(n)$ are given by:

$$\begin{aligned} \underline{\hat{x}}(n) &= [f(x(n))\, f(x(n-1)) \ldots f(x(n-N+1))]^T \\ \hat{d}(n) &= g(d(n)) \end{aligned} \tag{2.19}$$

where f and g are memoryless odd-symmetric nonlinear functions.

As it is evident from the structure of the proposed adaptive configuration, NLMS algorithm is based on the conventional LMS algorithm but incorporates nonlinear transformations of the input and the desired signals. The primary function of the nonlinearity is to decrease the eigenvalue spread of the input autocorrelation matrix. Later, we will show how to use a proper nonlinearity

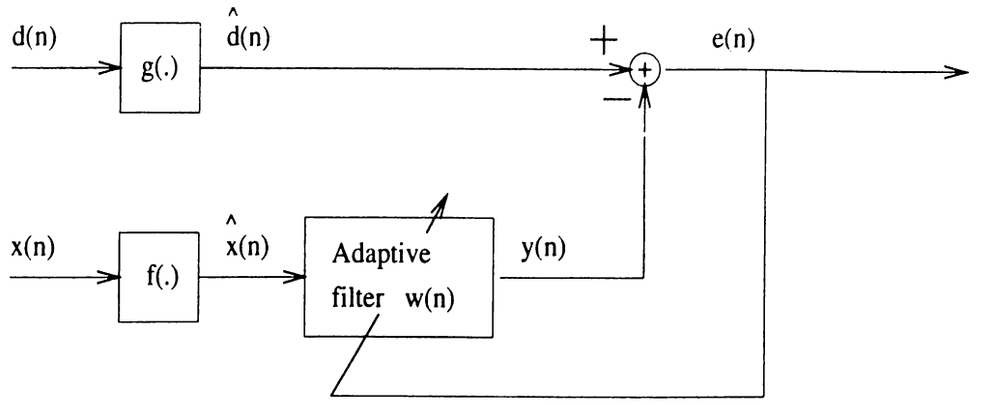

Figure 2.2: *NLMS block diagram*

to filter α-stable non-Gaussian processes, and processes which are corrupted by additive impulsive interference.

In this chapter we consider only the case in which $f = g$. We found the following so-called *softlimiter* nonlinearity most useful in our studies:

$$f(x) = \begin{cases} \alpha & \text{if } x \geq \alpha \\ x & \text{if } -\alpha < x < \alpha \\ -\alpha & \text{if } x \leq -\alpha \end{cases}$$

Analysis of the effects of the softlimiting on the correlation function of the input data is difficult, and analytical results are hard to obtain. We have explicitly evaluated the autocorrelation function of a softlimited correlated Gaussian process by approximating the softlimiter with a continuous differentiable function. According to the results presented in Section 2.4, softlimiting a correlated Gaussian process causes a decrease in the autocorrelation coefficients:

$$\hat{\rho}(m) < \rho(m) \; ; \quad m = 1, 2, \ldots \tag{2.20}$$

Hence, the softlimiting can be viewed as some kind of a decorrelation operation which flattens the spectrum of the input process. Since the ratio of the maximum eigenvalue to the minimum eigenvalue is bounded by the ratio of the largest to the smallest component in the power spectrum [2], the previous statement implies that the condition number of R_{xx} can actually be decreased by such a transformation. While this explanation may not be valid for all types of input data and cannot be taken as a proof, simulations have shown that this

is the case for all but a few exceptional cases. The following examples are presented to confirm the credibility of the conclusions drawn in this section.

Example 1. Consider a first-order auto regressive process (AR(1)):

$$x(n) = ax(n-1) + u(n) \tag{2.21}$$

where $u(n)$ is a random zero mean i.i.d. Gaussian process with variance σ_u^2. It is well known that the normalized autocorrelation function of this process is:

$$\rho(m) = a^m \quad . \tag{2.22}$$

Therefore R_{xx} at order N takes the following form

$$R_{xx} = \begin{bmatrix} 1 & a & a^2 & \cdots & a^{N-1} \\ a & 1 & a & \cdots & a^{N-2} \\ & & & & \\ a^{N-1} & a^{N-2} & \cdots & a & 1 \end{bmatrix} \tag{2.23}$$

The eigenvalue spread of R_{xx}, $\chi(R_{xx}) = \frac{\lambda_{max}}{\lambda_{min}}$, increases as $a \rightarrow 1$ and it is a monotonic nondecreasing function of the order of the matrix N. The case for which $a = 0$ corresponds to the input being white , $\chi(R_{xx}) = 1$, and we obtain the highest possible convergence rate for LMS adaptation.

Let us take 2000 samples of an AR(1) process with $a = 0.99$ and $\sigma_u^2 = 1 - 0.99^2$. The process was passed through a softlimiter and the variation of the condition number is plotted in Fig. 2.3.a as a function of the clipping value of the softlimiter as it varies from $3\sigma_x$ to $0.01\sigma_x$. For $\alpha = 3$, $\chi(R_{xx}) = 626$ and for $\alpha = 1$, $\chi(R_{xx}) = 456$. Using NLMS with $\alpha = 1$ instead of pure LMS in this case would speed up the convergence of the filter coefficients by a factor of roughly equal to 1.37.

Example 2: Consider a signal $x(n)$ which is obtained by passing a zero mean white Gaussian process through a 32^{nd} order linear phase FIR noise coloring filter. The frequency response of the coloring filter is given in Fig. 3.3.a. $x(n)$ models a correlated Gaussian process whose autocorrelation sequence is the autocorrelation of the filter impulse response since the input to the coloring filter is white. Eigenvalue spread of R_{xx} at order $N = 7$ is $\chi(R_{xx}) = 2096$.

Again, 2000 samples of $x(n)$ passed through a softlimiter with varying clipping value were used to estimate the condition number of the process at order 7.

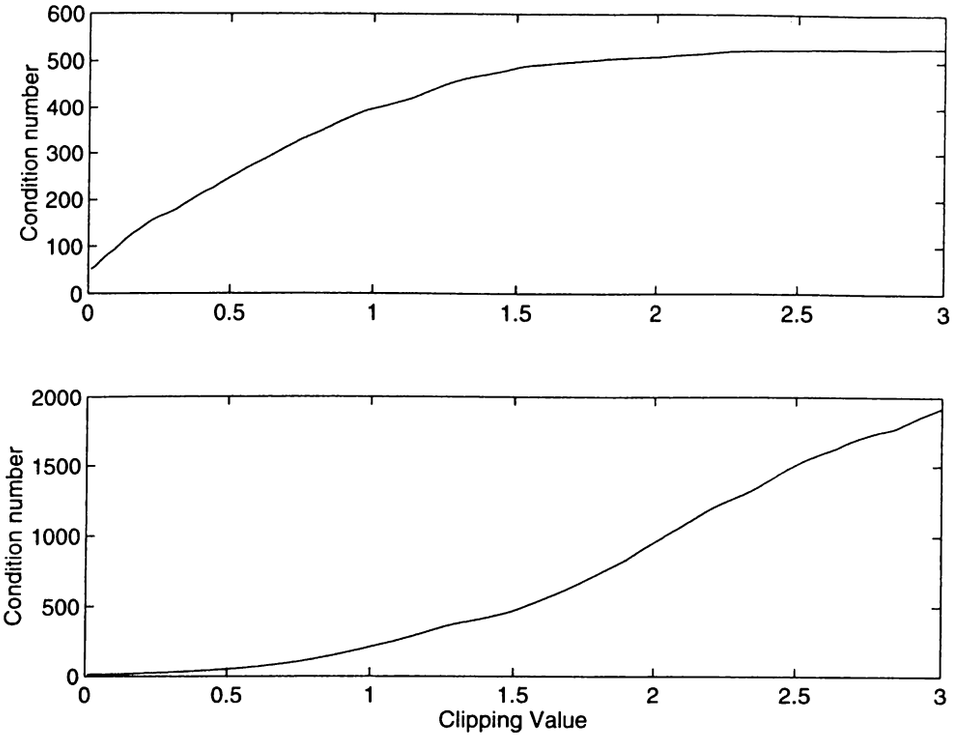

Figure 2.3: *Variation of condition number as a function of clipping value of the softlimiter. Upper curve corresponds to the AR(1) process, and the lower curve corresponds to the colored noise.*

The variation of the condition number is plotted as a function of *alpha* in Fig. 2.3.b . The plot shows that the condition number decreases with decreasing clipping, which confirms our intuitive explanation.

The produced signal $x(n)$ was used to perform a system identification experiment, where the desired signal was derived by passing the input through a 7^{th} order FIR filter $\underline{h} = [1\ 0\ \ldots\ 0]^T$, which is the model system. At convergence coefficients of the adaptive filter may be taken to be the estimates of those of the model system, hence the name system identification. Both the LMS and the NLMS algorithms were used in the adaptation and the convergence of the first tap weight of the adaptive filter is plotted in Fig. 2.4, by averaging the results of 100 simulations. The upper curve corresponds to the lowest value of the clipping and the clipping value decrease as we move down the curves. The slowest converging curve belongs to LMS. The stepsizes of LMS and NLMS

were adjusted to have the maximum possible rate of convergence without encountering the danger of instability. As observed, it is possible to obtain much higher convergence rates using the NLMS algorithm in the adaptation.

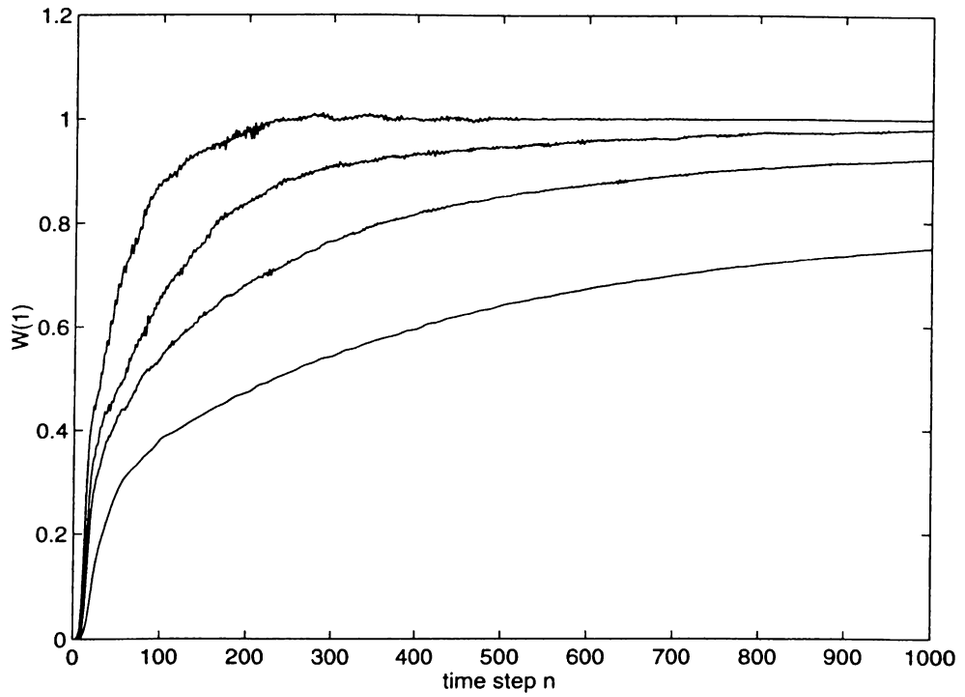

Figure 2.4: *Convergence curve of the first tap weight for different degrees of clipping.*

2.4 Convergence Analysis

So far we have only dealt with the implications of softlimiting on the eigenvalue spread of the input autocorrelation matrix, and we have seen that such a transformation could be used to decrease $\chi(R_{xx})$. On the other hand, applying a softlimiting operation to the input and the desired signals certainly changes the underlying statistics of these signals. Therefore we cannot expect the NLMS algorithm to converge to the same solution provided by the LMS.

In this section we will try to derive expressions for the mean weight provided by the NLMS algorithm at convergence. The analysis is based on computing

the desired statistics of the softlimited processes by approximating the softlimiter with a continuous differentiable function.

In order to simplify the analysis, we will use the following analytic function to approximate the soft nonlinearity

$$f(x) = \sqrt{\frac{2}{\pi}} \frac{1}{\sigma} \int_0^x e^{-z^2/2\sigma^2} \, dz \quad . \tag{2.24}$$

Here, the scalar parameter σ controls the sharpness of the nonlinear function at the origin and the degree of saturation. The behavior of $f(x)$ can be varied by changing σ. For example

$$\lim_{\sigma \to 0} f(x) = \text{sign}[x] \tag{2.25}$$

and

$$\lim_{\sigma \to \infty} f(x) = \sqrt{\frac{2}{\pi}} \frac{x}{\sigma} \tag{2.26}$$

Therefore the hard limiter and the linear function are limiting cases of (2.24).

We assume that the input signal is a stationary Gaussian process with zero mean and variance σ_x^2. i.e.

$$E[x(n-i)x(n-j)] = \mu_{ij} \quad i \neq j \quad . \tag{2.27}$$

We will assume that the following relationship exists between the input and the desired signals:

$$d(n) = \underline{h}^T \underline{x}(n) \tag{2.28}$$

Inserting this into the NLMS update we get

$$\underline{w}(n+1) = \underline{w}(n) + \mu \left[f(\underline{h}^T \underline{x}(n)) - \underline{w}^T(n) f(\underline{x}(n)) \right] f(\underline{x}(n)) \tag{2.29}$$

In this case, the parameter error vector is

$$\underline{v}(n) = \underline{w}(n) - \underline{h} \tag{2.30}$$

By writing the NLMS update in terms of the parameter error, vector we get

$$\begin{aligned} \underline{v}(n+1) \;=\; & \underline{v}(n) + \mu f(\underline{h}^T \underline{x}(n)) f(\underline{x}(n)) - \mu f(\underline{x}(n)) f(\underline{x}^T(n)) \underline{v} \\ & - \mu f(\underline{x}(n)) f(\underline{x}^T(n)) \underline{h} \end{aligned} \tag{2.31}$$

Taking the expectation of (2.31) and assuming that the parameter error vector $\underline{v}(n)$ and the data vector $\underline{x}(n)$ are independent of each other we obtain

$$\begin{aligned} E[\underline{v}(n+1)] &= E[\underline{v}(n)] + \mu E[f(\underline{h}^T\underline{x}(n))f(\underline{x}(n))] \\ &\quad -\mu E[f(\underline{x}(n))f(\underline{x}^T(n))]E[\underline{v}(n)] - \mu E[f(\underline{x}(n))f(\underline{x}^T(n))]E[\underline{h}] \end{aligned} \tag{2.32}$$

Thus the right hand side (RHS) of (2.32) requires the evaluation of the nonlinear Gaussian functionals of the form

$$E[f(\underline{h}^T\underline{x}(n))f(x(n-i))]; \quad i = 0, 1, \ldots, N-1 \tag{2.33}$$

and

$$E[f(x(n-i))f(x^T(n-j))]; \quad i, j = 0, 1, \ldots, N-1 \ . \tag{2.34}$$

The desired expectation involves nonlinear functionals of a pair Gaussian variates which can be evaluated by using Price's theorem, which is stated in the following paragraph:

Price's Theorem: Given two jointly normal RVs x and y, we form the mean

$$I = E[g(x,y)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y)f(x,y)\,dx\,dy \ . \tag{2.35}$$

of some function $g(x,y)$ of (x,y). The above integral is a function $I(\mu)$ of the covariance μ of the RVs x and y. Then

$$\frac{\partial^n I(\mu)}{\partial \mu^n} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{\partial^{2n} g(x,y)}{\partial x^n \partial y^n} f(x,y)\,dx\,dy = E\left[\frac{\partial^{2n} g(x,y)}{\partial x^n \partial y^n}\right] \tag{2.36}$$

The advantage of this approach is that it is often not possible to directly evaluate $E[g(x,y)]$. However, the expectations of the derivatives of $g(.)$ can usually be evaluated easily so that, the desired expectation can be formed by integrating over μ.

By using this theorem, we first evaluate $E[f(x(n-i))f(x(n-j))]$. Let $x_i = x(n-i)$ and $x_j = x(n-j)$. x_i and x_j are jointly normal with zero mean and covariance $E[x_i x_j] = \mu_{ij}$. Applying Price's theorem to the unknown expectation

$$\begin{aligned} \frac{d\,I(\mu)}{d\,\mu} &= E\left[\frac{d\,f(x_i)}{d\,x_i}\frac{d\,f(x_j)}{d\,x_j}\right] \\ &= E[\frac{2}{\pi}\frac{1}{\sigma^2}e^{-x_i^2/2\sigma^2}e^{-x_j^2/2\sigma^2}] \\ &= \frac{2}{\pi}\frac{1}{\sigma^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\frac{1}{2\pi \mid M \mid^{1/2}}e^{-\frac{x_i^2+x_j^2}{2\sigma^2}}e^{\{-\frac{1}{2}\underline{x}^T M^{-1}\underline{x}\}}d\underline{x} \end{aligned} \tag{2.37}$$

where $\underline{x} = [x_i \; x_j]^T$,

$$M^{-1} = \begin{bmatrix} \sigma_x^2 & \mu \\ \mu & \sigma_x^2 \end{bmatrix}^{-1} = \frac{1}{\sigma_x^4 - \mu^2} \begin{bmatrix} \sigma_x^2 & -\mu \\ -\mu & \sigma_x^2 \end{bmatrix} \tag{2.38}$$

and $\mid M \mid = det(M) = \sigma_x^4 - \mu^2$. The integral in (2.37) can be put into the following form

$$\frac{d\,I(\mu)}{d\,\mu} = \frac{2}{\pi}\frac{1}{\sigma^2}\frac{\mid K \mid^{1/2}}{\mid M \mid^{1/2}} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{1}{2\pi \mid K \mid^{1/2}} e^{\{-\frac{1}{2}\underline{x}^T K^{-1}\underline{x}\}} d\underline{x} \tag{2.39}$$

where

$$K^{-1} = \begin{bmatrix} \frac{\sigma_x^2}{\sigma_x^4-\mu^2} + \frac{1}{\sigma^2} & \frac{-\mu}{\sigma_x^4-\mu^2} \\ \frac{-\mu}{\sigma_x^4-\mu^2} & \frac{\sigma_x^2}{\sigma_x^4-\mu^2} + \frac{1}{\sigma^2} \end{bmatrix} \tag{2.40}$$

and

$$\mid K \mid = det(K) = \frac{1}{det(K^{-1})} = \frac{1}{\sigma_x^4 - \mu^2} + 2\frac{\sigma_x^2}{\sigma^2}\frac{1}{\sigma_x^4 - \mu^2} + \frac{1}{\sigma^4} \tag{2.41}$$

The integral in (2.42) is equal to one. Therefore we have the following result

$$\begin{aligned} \frac{d\,I(\mu)}{d\,\mu} &= \frac{2}{\pi}\frac{1}{\sigma^2}\frac{\mid K \mid^{1/2}}{\mid M \mid^{1/2}} = \frac{2}{\pi}\frac{1}{\sigma^2}\frac{1}{\sqrt{\sigma_x^4 - \mu^2}}\frac{1}{\sqrt{\frac{1}{\sigma_x^4-\mu^2} + \frac{1}{\sigma^4} + 2\frac{\sigma_x^2}{\sigma^2}\frac{1}{\sigma_x^4-\mu^2}}} \\ &= \frac{2}{\pi}\frac{1}{\sqrt{\eta^2 - \mu^2}} \end{aligned} \tag{2.42}$$

where $\eta^2 = \sigma^4 + 2\sigma_x^2\sigma^2 + \sigma_x^4$. By integrating the above expression with respect to μ and putting the initial condition $I(0) = 0$, we arrive at the following expression

$$I(\mu) = \frac{2}{\pi}\sin^{-1}\left(\frac{\mu}{\eta}\right) = \frac{2}{\pi}\sin^{-1}\left(\frac{\mu}{\sigma^2 + \sigma_x^2}\right) \tag{2.43}$$

Next, we need to evaluate $E[f(\underline{h}^T\underline{x}(n))f(x(n-i))]$. The desired response $d(n) = \underline{h}^T\underline{x}(n)$ and the components of $\underline{x}(n)$ are jointly distributed Gaussian variates with zero means and with variances and covariances given by

$$\mu_i = E[\underline{h}^T\underline{x}(n)x(n-i)] = \underline{h}^T\underline{r}(i) \tag{2.44}$$

$$E[d^2(n)] = \underline{h}^T E[\underline{x}^T(n)\underline{x}(n)]\underline{h} \tag{2.45}$$

$$= \underline{h}^T R_{xx}\underline{h} \tag{2.46}$$

$$E[x^2(n-i)] = \sigma_x^2 \tag{2.47}$$

where

$$\underline{r}(i) = [r(i)\, r(i+1)\, \ldots\, r(i+N-1)] \tag{2.48}$$

$$r(i) = E[x(n)x(n-i)] \tag{2.49}$$

Thus we see that the form of the expectation is the same as the one in (2.34). By carrying out the same steps from (2.37) to (2.43) we obtain the following result

$$E[f(\underline{h}^T\underline{x}(n))f(x(n-i))] = \frac{2}{\pi}\sin^{-1}\left(\frac{\underline{h}^T\underline{r}(i)}{\sigma^2 + \sigma_x\underline{h}^T R_{xx}\underline{h}}\right) \tag{2.50}$$

Therefore the input autocorrelation matrix and the vector of crosscorrelations between the input and the desired signals are given by

$$\hat{R}_{xx} = \left\{\frac{2}{\pi}\sin^{-1}\left(\frac{E[x(n-i)x(n-j)]}{\sigma^2+\sigma_x^2}\right)\right\}_{i,j} \tag{2.51}$$

$$\hat{\underline{r}}_{xd} = \left\{\frac{2}{\pi}\sin^{-1}\left(\frac{\underline{h}^T\underline{r}(i)}{\sigma^2 + \sigma_x\underline{h}^T R_{xx}\underline{h}}\right)\right\}_{i} \tag{2.52}$$

Substituting (2.51) and (2.52) in to (2.32) yields

$$\underline{v}(n+1) = (I - \mu\hat{R}_{xx})\underline{v}(n) + \mu(\hat{\underline{r}}_{xd} - \hat{R}_{xx}\underline{h}) \tag{2.53}$$

Hence, we find that the parameter error vector $\underline{v}(n)$ converges in the mean to

$$E[\underline{v}(n+1)] \rightarrow \hat{R}_{xx}^{-1}\hat{\underline{r}}_{xd} - \underline{h} \tag{2.54}$$

provided that the algorithm is stable. It is assumed that the modified input autocorrelation matrix $\hat{R}_{xx}$ is invertible.

The expression found for the bias of the parameter error vector at convergence is not sufficient for describing the convergence properties of the NLMS algorithm. It is necessary to describe the time evolution of the parameter error vector and find an expression for the stability range of the algorithm in terms of the adaptation constant. In order to do this, we define a new parameter error vector as

$$\hat{\underline{v}}(n) = E[\underline{v}(n)] - \hat{R}_{xx}^{-1}\hat{\underline{r}}_{xd} + \underline{h} \ . \tag{2.55}$$

We can rewrite (2.31) in terms of $\hat{\underline{v}}(n)$ as

$$\hat{\underline{v}}(n+1) = (I - \mu\hat{R}_{xx})\hat{\underline{v}}(n) \ . \tag{2.56}$$

We now have a recursive equation which describes the evolution of the error in NLMS coefficients. First, we try to decouple the update equations by diagonalizing positive definite symmetric matrix $\hat{R}_{xx}$. $\hat{R}_{xx}$ can be decomposed as

$$\hat{R}_{xx} = \hat{Q}^T \hat{\Lambda} \hat{Q} \tag{2.57}$$

where $\hat{Q}$ is the modal matrix which has normalized eigenvectors of $\hat{R}_{xx}$ as its columns

$$\hat{Q} = [\underline{\hat{q}}_1 \; \underline{\hat{q}}_2 \; \cdots \; \underline{\hat{q}}_N], \tag{2.58}$$

$$\hat{Q}^T \hat{Q} = I \tag{2.59}$$

where $\underline{\hat{q}}_i$ is the eigenvector corresponding to $\hat{\lambda}_i$. $\hat{\Lambda}$, the so called spectral matrix, is a diagonal matrix whose diagonal elements consist of the eigenvalues of $\hat{R}_{xx}$

$$\hat{\Lambda} = \begin{bmatrix} \hat{\lambda}_1 & 0 & & 0 \\ 0 & \hat{\lambda}_2 & . & \\ & 0 & & \\ . & & & 0 \\ 0 & 0 & . \; 0 & \hat{\lambda}_N \end{bmatrix} \tag{2.60}$$

Returning to equation (2.56) and substituting (2.57) we get:

$$\underline{\hat{v}}(n+1) = (I - \mu \hat{Q}^T \Lambda \hat{Q}) \underline{\hat{v}}(n) \tag{2.61}$$

The matrix $\hat{Q}$ is now used to defined the rotated error vector $\underline{u}(n+1)$

$$\underline{u}(n) = \hat{Q} \underline{\hat{v}}(n) \quad . \tag{2.62}$$

Multiplying both sides of equation by $\hat{Q}$ and using the fact that $\hat{Q}^T \hat{Q} = I$ we arrive at the following equation

$$\underline{u}(n+1) = (I - \mu \hat{\Lambda}) \underline{u}(n+1) \tag{2.63}$$

In view of the diagonal nature of $\hat{\Lambda}$, we can separate decoupled equations for each element of $\underline{u}(n)$ as

$$\underline{u}_i(n+1) = (1 - \mu \hat{\lambda}_i) \underline{u}_i(n) \tag{2.64}$$

This first-order scalar difference equation can be solved by using repeated substitution, providing the following result

$$\underline{u}_i(n+1) = (1 - \mu\hat{\lambda}_i)^n \underline{u}_i(0) \quad . \tag{2.65}$$

Therefore, in the limit

$$\lim_{n\to\infty} \underline{u}_i(n+1) = 0 \tag{2.66}$$

provided that

$$0 < \mu < \frac{2}{\hat{\lambda}_i} \tag{2.67}$$

Hence, the condition for convergence is

$$0 < \mu < \frac{2}{\hat{\lambda}_{max}} \tag{2.68}$$

where $\hat{\lambda}_{max}$ is the largest eigenvalue of $\hat{R}_{xx}$. On the other hand, the time constant for the convergence of i^{th} mode can easily be calculated as

$$t_i \approx \frac{1}{\mu\hat{\lambda}_i} \quad . \tag{2.69}$$

The convergence rate of the NLMS algorithm is clearly limited by the mode which has the smallest eigenvalue λ_{min}. Combining equations (2.68) and (2.69) we have the following expression for the time constant of NLMS algorithm:

$$t \approx \frac{\hat{\lambda}_{\text{max}}}{\hat{\lambda}_{\text{min}}} \quad . \tag{2.70}$$

Therefore we see that the NLMS algorithm converges in a time proportional to the eigenvalue spread of the autocorrelation matrix of the softlimited input data. In section 2.3, we intuitively argued that passing input through a softlimiter decreases the eigenvalue spread. Therefore, based on this argument, we can say that $t_{NLMS} < t_{LMS}$.

The above analysis has been carried out for $f(.) = g(.)$ and in particular for f being a softlimiter. Under these restrictions it is seen that the algorithm is biased. An interesting question is whether we can remove the bias entirely by using a different structure for the AF configuration. If $\hat{d}(n) = \underline{h}^T f(\underline{x}(n))$, the RHS of (2.57) becomes

$$\hat{R}_{xx}^{-1}\hat{\underline{r}}_{xd} - \underline{h} = \hat{R}_{xx}^{-1}\hat{R}_{xx}\underline{h} - \underline{h} = \underline{0} \tag{2.71}$$

and NLMS is an asymptotically unbiased estimator. Furthermore the evolution of the parameter error vector is governed by the properties of the modified input autocorrelation matrix $\hat{R}_{xx}$. Thus, NLMS converges **much faster** to the optimal Wiener solution if a suitable structure can be found to have $\hat{d}(n) = \underline{h}^T f(\underline{x}(n))$. This kind of an exact relation between the input and the desired signals can be obtained in system identification applications where the input to the plant is *nonwhite*[42]. The modified NLMS structure is schematically represented in Fig. 2.5.

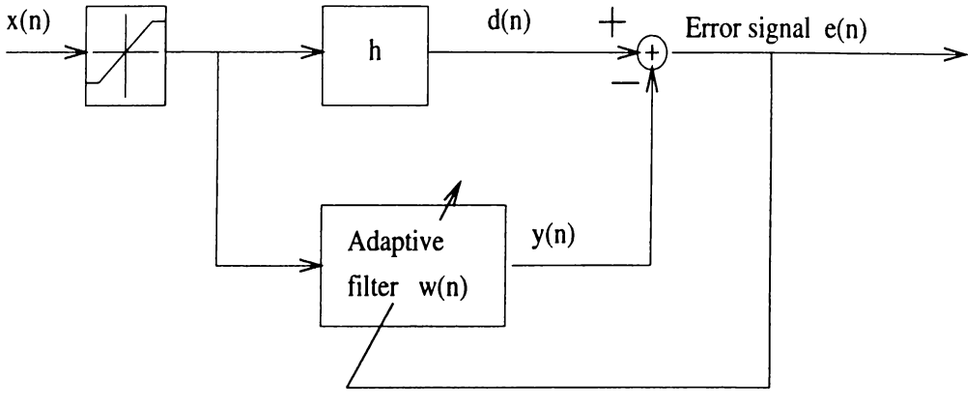

Figure 2.5: *Modified NLMS structure for use in system identification*

2.5 Computer Simulation

Let $x(k) = a_1 x(k-1) + a_2 x(k-2) + u(n)$ be an $AR(2)$ process, where $a_1 = 1.9114$ and $a_2 = -0.95$ and $\{u(k)\}$ is an i.i.d. white noise sequence. The input signal to the adaptive filter was $x(n)$, and the desired signal was $d(n) = x(n+1)$. In Fig.2.6 transient behavior of the tap weights provided by the LMS and NLMS adaptation algorithms are plotted. The step size $\mu_{LMS} = 0.11$ was adjusted so that the LMS adaptation was at the edge of instability. The step size of the NLMS adaptation was $\mu_{NLMS} = 0.4$ and the input and the desired signals were clipped at $\alpha = \mp 1.5$. The plot shows improved convergence rate of the NLMS relative to the LMS. The final values of the tap weights of the NLMS and the LMS are different due to the bias introduced by the nonlinear transformation of the input signals.

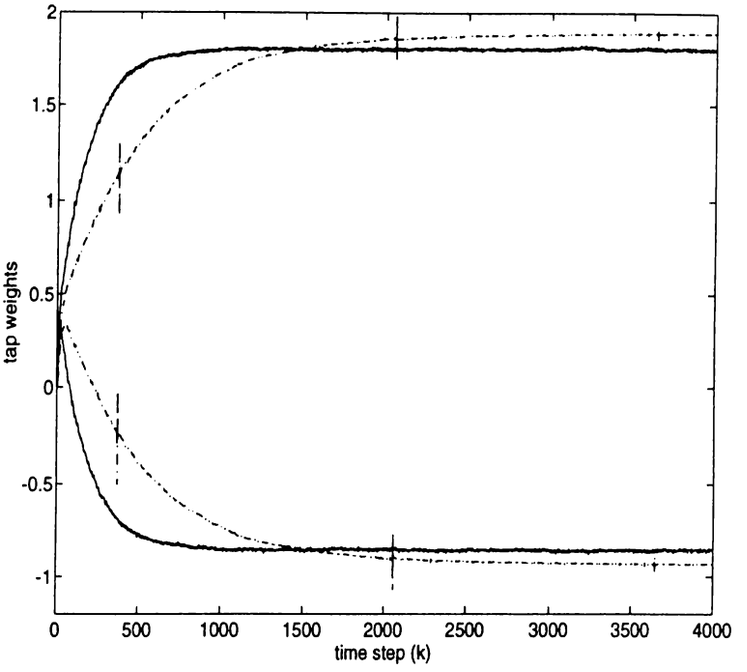

Figure 2.6: *Transient behavior of tap weights. Solid line* $\underline{w}_{NLMS}$*, dashed line* $\underline{w}_{LMS}$.

2.6 Steady State Mean Square Error

In the previous section, we derived expressions for the mean weight at convergence of the NLMS algorithm, and obtained bounds for the stability range of the algorithm. However, the analysis of the convergence behavior in the mean is of limited value without obtaining results for the behavior of the system about the steady-state solution. In this section, we will be concerned with the analysis of the mean square error $\hat{J}(n)$ and misadjustment for the adaptive filter.

As described in section (2.2.2), the mean square error can be written as

$$\hat{J}(n) = \hat{J}_{\min} + \underline{\hat{v}}^T(n)\hat{R}_{xx}\underline{\hat{v}}(n) \tag{2.72}$$

where $\underline{\hat{v}}(n) = \underline{w}(n) - \hat{R}_{xx}^{-1}\hat{r}_{xd}$, is the parameter error vector as defined in section (2.4). Using (2.57) and (2.62), (2.71) can be put into the following form

$$\hat{J}(n) = \hat{J}_{\min} + \underline{u}^T(n)\Lambda\underline{u}(n) \tag{2.73}$$

where $\underline{u}(n) = \hat{Q}\underline{\hat{v}}(n)$ and $\hat{R}_{xx} = \hat{Q}^T\hat{\Lambda}\hat{Q}$. The elements of the diagonal of $\hat{\Lambda}$ are the eigenvalues of the autocorrelation matrix. Equation (2.71) can be rewritten

as

$$\hat{J}(n) = \hat{J}_{\min} + \sum_{k=1}^{N} \hat{\lambda}_k u_k^2(n) \tag{2.74}$$

where $u_k^2(n)$ is the k^{th} element of the vector $\underline{u}(n)$. Taking the expectation of both sides

$$E[\hat{J}(n)] = \hat{J}_{\min} + \sum_{k=1}^{N} \hat{\lambda}_k E[u_k^2(n)] \tag{2.75}$$

From now on we will assume that the convergence of the algorithm has taken place in the steady state. Steady-state value of $\hat{J}(n)$ will be denoted as $\hat{J}_\infty$. The NLMS algorithm uses an estimate of the gradient at each iteration [1]-[2]. The update equation can be written as

$$\underline{w}(n+1) = \underline{w}(n) - \mu \hat{\underline{\nabla}}(n) \tag{2.76}$$

where $\hat{\underline{\nabla}}$ is the estimate of the gradient. It can be written as

$$\hat{\underline{\nabla}}(n) = \underline{\nabla}(n) + \underline{\varepsilon}(n) \tag{2.77}$$

where $\underline{\nabla}(n)$ is the true gradient and $\underline{\varepsilon}(n)$ is the noise of the estimate. Substituting (2.77) into LMS update and using $\underline{\nabla}(n) = 2\hat{R}_{xx}\underline{w}(n) - 2\hat{r}_{xd}$ [7] we have

$$\underline{\hat{v}}(n+1) = 2\left(I - \mu\hat{R}_{xx}\right)\underline{\hat{v}}(n) - \mu\underline{\varepsilon}(n) \tag{2.78}$$

Multiplying both sides of (2.78) by Q

$$\underline{u}(n+1) = 2\left(I - \mu\hat{\Lambda}\right)\underline{u}(n) - \mu\underline{\varepsilon}(n) \tag{2.79}$$

Assuming wide-sense stationarity at convergence and taking the covariance of both sides

$$4(\Lambda - \mu\Lambda^2)\text{cov}[\underline{u}(n)] = \mu\text{cov}[\underline{\varepsilon}(n)] \tag{2.80}$$

Since at convergence the exact gradient is zero

$$[\underline{\varepsilon}(n)] = -2e(n)\underline{x}(n) \quad . \tag{2.81}$$

At convergence the error is orthogonal to the input vector. Thus, we get

$$\begin{aligned} \text{cov}[\underline{\varepsilon}(n)] &= Q^T\text{cov}[\underline{\varepsilon}(n)]Q & (2.82)\\ &= 4Q^T\hat{R}_{xx}QE[e^2(n)] & (2.83)\\ &= 4\Lambda E[e^2(n)] \quad . & (2.84) \end{aligned}$$

Since, $E[e^2(n)]$ converges to the steady-state MSE $\hat{J}_\infty$

$$\text{cov}[\underline{\varepsilon}(n)] = 4\Lambda\hat{J}_\infty \quad . \tag{2.85}$$

By substituting (2.85) into (2.82) we obtain

$$\begin{aligned} E[\underline{u}_i^2(n)] &= \{\text{cov}[\underline{u}(n)]\}_{i,i} && (2.86)\\ &= \left\{\mu\hat{J}_\infty(\Lambda - \mu\Lambda^2)^{-1}\Lambda\right\}_{i,i} && (2.87)\\ &= \mu\hat{J}_\infty\left(\frac{1}{1-\mu\lambda_i}\right) && (2.88) \end{aligned}$$

Now, by using (2.75) we obtain

$$\hat{J}_\infty = \lim_{n\to\infty} E[\hat{J}(n)] = \hat{J}_{\text{min}} + \sum_{i=1}^{N} \mu\hat{J}_\infty\left(\frac{1}{1-\mu\hat{\lambda}_i}\right) \tag{2.89}$$

For this case, the misadjustment is

$$M = \frac{\hat{J}_\infty - \hat{J}_{\text{min}}}{\hat{J}_{\text{min}}} \quad , \tag{2.90}$$

By using (2.89), the corresponding misadjustment is found as

$$\begin{aligned} M &= \frac{\pi}{1-\pi}; && (2.91)\\ \pi &= \sum_{i=1}^{N}\left(\frac{\mu\hat{\lambda}_i}{1-\mu\hat{\lambda}_i}\right) && (2.92) \end{aligned}$$

By using

$$\sum_{i=1}^{N} \mu\lambda_i = \mu\text{tr}\hat{R}_{xx} \tag{2.93}$$

we find that the misadjustment is given by

$$M = \frac{\mu\text{tr}\hat{R}_{xx}}{1-\mu\text{tr}\hat{R}_{xx}} \tag{2.94}$$

2.7 Performance of NLMS Under Additive Impulsive Interference

It has been known that the LMS algorithm may degrade unacceptably when the input signals are corrupted by impulsive interference [7]. Sparse impulses

arise in a variety of practical applications such as speech, biomedical and image processing. The proposed algorithm suggests an improvement in the performance over the LMS in such situations.

The new configuration can be useful in applications where additive impulsive interference is a problem. The nonlinear transformation of the input signals protects the filter coefficients from the impact of impulsive interferences by limiting the magnitude of the sparse impulses. Unlike other LMS variants developed to cope with the impulsive interference problem, the proposed method does not introduce additional computational effort.

Consider an adaptive filtering configuration where the input and the desired signals are corrupted by impulsive interferences $\xi(n)$ and $\eta(n)$ respectively.

$$x'(n) = x(n) + \xi(n) \tag{2.95}$$

$$d'(n) = d(n) + \eta(n) \ . \tag{2.96}$$

Here we assume that the impulsive interferences $\xi(n)$ and $\eta(n)$ are zero mean i.i.d. sequences which are mutually independent and are independent of both $x(n)$ and $d(n)$. We also assume that the input sequence is i.i.d. with $\sigma_x^2 = 1$. We assume that the following relation exists between the desired signal and the input signal:

$$d(n) = \underline{h}^t \underline{x}(n) \ . \tag{2.97}$$

We will examine three different cases

1. $\xi(n) = 0$ $\eta(n) = 0$ for all n.

2. $\xi(n) = 0$ for all n.

3. $\eta(n) = 0$ for all n.

Throughout the analysis the impulsive signals will be assumed to have the following form

$$\zeta(n) = c(n)A(n) \tag{2.98}$$

where $c(n)$ is an i.i.d. sequence with

$$P\{c(n) = 1\} = a; \qquad P\{c(n) = 0\} = 1 - a \tag{2.99}$$

where a is the arrival probability. The distribution of the amplitude $A(n)$ is arbitrary subject to the constraint that $var\{A(n)\} \gg \sigma_x^2$. We will assume that the amplitude is independent of the arrival time.

Case I. In this case the relation between the input and the desired signals is exact and LMS algorithm converges to the following optimal set of coefficients provided that the algorithm is stable[2]

$$E[\underline{w}(n)] = R_{xx}^{-1}\underline{r}_{xd} \tag{2.100}$$

where $R_{xx} = E[\underline{x}\ \underline{x}^t]$ and $\underline{r}_{xd} = E[\underline{x}d]$.
Case II. In this case input signals are

$$d'(n) = d(n) + \eta(n) \tag{2.101}$$

and

$$x'(n) = x(n) \tag{2.102}$$

Defining the parameter error vector as $\underline{v}(n) = \underline{w}(n) - \underline{h}$, the LMS update can be written as

$$\underline{v}(n+1) = (I - \mu\underline{x}(n)\underline{x}^t(n))\underline{v}(n) + \mu\underline{x}(n)\eta(n) \quad . \tag{2.103}$$

Taking the expectations and assuming that the input signal $x(n)$ is independent of the parameter error vector $v(n)$, we get

$$E[\underline{v}(n+1)] = (I - \mu E[\underline{x}(n)\underline{x}^t(n)])E[\underline{v}(n)] \quad . \tag{2.104}$$

With $x(n)$ an i.i.d. sequence as assumed, $E[\underline{x}(n)\underline{x}^t(n)] = \sigma_x^2$ and the parameter estimates are in the mean decoupled. This means

$$E[v_i(n+1)] = (1 - \mu\sigma_x^2)E[v_i(n)] \quad i = 1, 2, \ldots, N \tag{2.105}$$

If we choose $0 < \mu < 2/\sigma_x^2$ the algorithm converges to

$$\underline{w}_\infty = R_{xx}^{-1}\underline{r}_{xd} = \underline{h} \quad . \tag{2.106}$$

Hence, in this case the parameter estimates are unbiased as in Case I. However, their variances are not smaller than those in Case I.
Case III. In this case we have

$$d'(n) = d(n) \tag{2.107}$$

and

$$x'(n) = x(n) + \xi(n) \tag{2.108}$$

The parameter error equation becomes

$$\underline{v}(n+1) = \underline{v}(n)+\mu \left[\underline{x}^t(n)\underline{h} - (\underline{x}(n) + \underline{\xi}(n))^t\underline{v}(n) - (\underline{x}(n) + \underline{\xi}(n))^t\underline{h}\right] (\underline{x}(n)+\underline{\xi}(n)) \tag{2.109}$$

Taking expectations we obtain

$$E[\underline{v}(n+1)] = I - \mu(E[\underline{x}(n)\underline{x}^t(n)] + E[\underline{\xi}(n)\underline{\xi}^t(n)])E[\underline{v}(n)] - \mu E[\underline{\xi}(n)\underline{\xi}^t(n)]\underline{h} \tag{2.110}$$

The above equation decouples to give for the i^{th} component of the parameter error vector

$$E[v_i(n+1)] = [1 - \mu(\sigma_x^2 + \sigma_\xi^2)]E[v_i(n)] - \mu\sigma_\xi^2 h_i \tag{2.111}$$

Thus LMS algorithm converges to

$$\underline{w}_\infty = \frac{\sigma_x^2}{\sigma_x^2 + \sigma_\xi^2}\underline{h} \tag{2.112}$$

Therefore in this case the LMS does not converge to the optimal parameter set and the bias is given by

$$E[\underline{v}(n+1)] \rightarrow -\frac{\sigma_\xi^2}{\sigma_x^2 + \sigma_\xi^2}\underline{h} \tag{2.113}$$

We see from (2.115) that the bias is proportional to the variance of additive impulsive noise. If we use a softlimiter which produces negligible distortion on the input and the desired signals but greatly decreases the variance of the impulsive noise component we can get a smaller bias. In other words, we want

$$f(x(n) + \xi(n)) \approx x(n) + f(\xi(n)), \tag{2.114}$$

$$g(d(n) + \eta(n)) \approx d(n) + g(\eta(n)) \tag{2.115}$$

where $f(.)$ and $g(.)$ are softlimiters whose clipping values are adjusted to yield minimal distortion on the input. Carrying out the analysis for the three cases considered above we obtain the following results:

Case I. The relation between the input and the desired signals are not affected and NLMS converges to the optimal set of coefficients as in (2.108).

Case II. Replacing $d(n)+\eta(n)$ by $d(n)+g(\eta(n))$ we see that NLMS converges to the same optimal Wiener solution as in (2.106) since $E[g(\eta(n))\underline{x}(n)] = 0$.

Case III. Replacing $x(n)+\xi(n)$ by $f(x(n)+\xi(n)) = x(n)+f(\xi(n))$ and retracing the analysis in (2.107) through (2.113), we obtain the following expression for the bias in the mean parameter error vector

$$E[\underline{v}(n+1)] \rightarrow -\frac{\hat{\sigma}_\xi^2}{\sigma_x^2 + \hat{\sigma}_\xi^2}\underline{h} \tag{2.116}$$

where

$$\hat{\sigma}_\xi^2 = E[f^2(\xi(n))]. \tag{2.117}$$

A convenient way is to set the clipping value of the softlimiter to three standard deviations above the mean of the input signals (assuming the signals are Gaussian distributed), since the effect of softlimiter on the correlation function of the input is practically absent for this value.

2.7.1 Computer Simulation

The potential of NLMS algorithm is illustrated in a simple adaptive filtering problem by considering the effects of impulsive interference. Let $x(n) = d(n)$ where $x(n)$ is an i.i.d. Gaussian input with $\sigma_x^2 = 0.01$. The impulsive interference has the form

$$\eta(n) = c_n A_n \tag{2.118}$$

where c_n is i.i.d. with

$$p(c_n = 1) = 0.02;\ p(c_n = 0) = 0.98 \tag{2.119}$$

The distribution of the amplitude A_n is Gaussian with $\text{var}\{A_k\} = 1$ and mean zero. The impulsive noise is added to the input signal $x(k)$. The transient behavior of tap weights for the NLMS and the LMS is plotted in Fig. 2.7 by averaging 50 independent trials. The parameters chosen for this simulation were: filter length $N = 2$, step size $\mu = 0.2$ and the clipping point of the soft-limiter $a = 0.2$. The LMS performed poorly in the presence of impulsive interference and failed to converge to the optimal solution $w^* = 1$. In contrast the NLMS algorithm exhibited a smooth convergence and a much smaller bias.

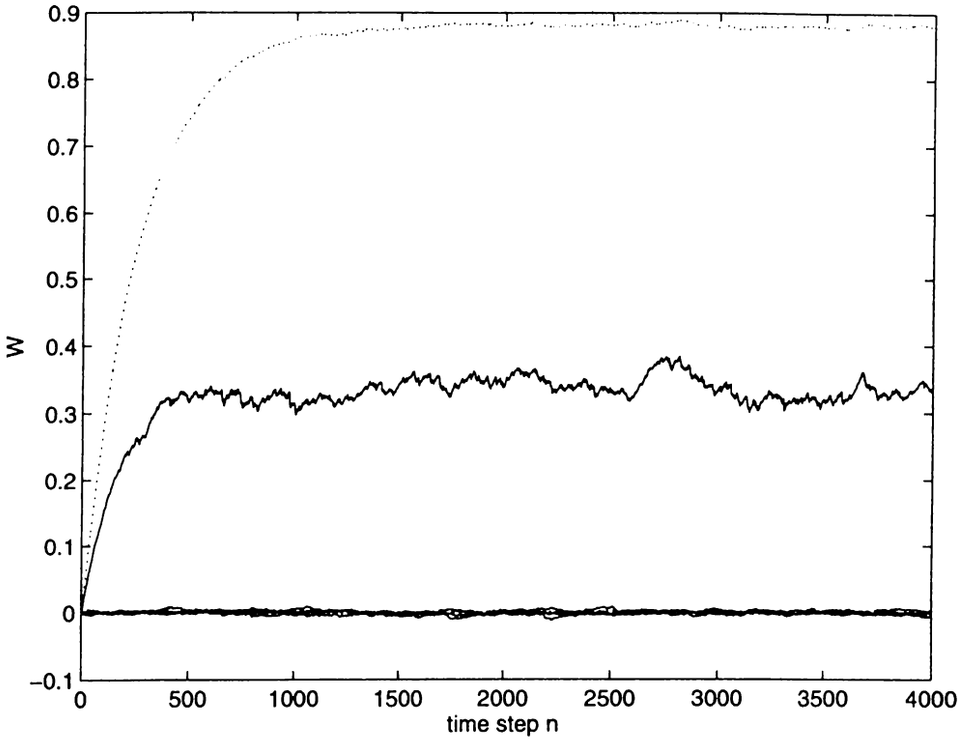

Figure 2.7: *Convergence curves of NLMS(dotted) and LMS(solid) under additive impulsive interference*

2.8 Adaptive Filtering Approaches for Non-Gaussian Stable Processes

Another application of our method is adaptive filtering for non-Gaussian stable processes [40]. There is a large class of physical phenomena which exhibits non-gaussian behavior such as underwater acoustic noise, low frequency atmospheric noise and many types of man-made noise. They exhibit sharp spikes and occasional bursts in their realizations. In the modeling of these type of signals and noise α-stable distributions can be used. Formally, a random variable is called α-stable if its characteristic function has the following form:

$$\phi(t) = \exp\{i\alpha t - \gamma|t|^{\alpha}[1 + i\beta \text{sign}(t)\omega(t,\alpha)]\} \tag{2.120}$$

where $-\infty < a < \infty$, $\gamma > 0$, $0 < \alpha \leq 2$, $-1 \leq \beta \leq 1$ and

$$\omega(t,\alpha) = \begin{cases} \tan(\alpha\pi/2) & \text{for } \alpha \neq 1 \\ \frac{2}{\pi}\log|t| & \text{for } \alpha = 1 \end{cases} \tag{2.121}$$

There is no explicit expression for the probability density function of these random variables except for $\alpha = 1$ and 2, which correspond to the Cauchy and Gaussian distributions, respectively. The *characteristic exponent* α is a measure of the deviation of the distribution from Gaussian. As α approaches zero the tails of the distribution function becomes heavier. For $1 \leq \alpha < 2$ the α-stable processes fail to have second or higher-order moments. As a result, many of the adaptive filtering algorithms which are based on the minimization of mean-square cost function become useless.

There are a number of solutions suggested to solve the minimization problem mentioned above with the motivation of gradient decent algorithms [28]-[40]. Such an algorithm, least mean p-norm (LMP) algorithm, is proposed in [40]. This algorithm is a generalization of instantaneous gradient descent algorithm to α-stable processes, where the gradient of the p-norm of the error,

$$\begin{aligned} J &= E[|e(n)|^p] \\ &= E[|d(n) - \underline{w}(n)'\,\underline{x}(n)|^p], \quad 0 < p < \alpha \end{aligned} \tag{2.122}$$

is used, and the tap weights, $\underline{w}$, are adapted at time step $n+1$ as follows:

$$\underline{w}(n+1) = \underline{w}(n) + \mu\ |e(n)|^{p-1}\text{sign}[e(n)]\ \underline{x}(n) \tag{2.123}$$

where μ is the step size which should be appropriately chosen. Note that, for $p = \alpha = 2$ the LMP algorithm reduces to the well-known LMS algorithm [1]. When p is chosen as 1, the LMP algorithm is called the Least Mean Absolute Deviation (LMAD) algorithm [40]:

$$\underline{w}(n+1) = \underline{w}(n) + \mu\text{sign}[e(n)]\ \underline{x}(n) \tag{2.124}$$

which is also known as the signed-LMS algorithm. With the motivation of the Normalized LMS algorithm, Normalized Least Mean p-Norm (NLMP) algorithm and Normalized Least Mean Absolute Deviation (NLMAD) algorithm were introduced in [28], with a superior performance than those of LMP and LMAD.

The normalized Least Mean p-Norm (NLMP) algorithm, uses the following update:

$$\underline{w}(n+1) = \underline{w}(n) + \beta \frac{|e(n)|^{p-1}\text{sign}(e(n))}{||\underline{x}(n)||_p^p + \lambda} \underline{x}(n) \tag{2.125}$$

where $\beta, \lambda > 0$ are appropriately chosen update parameters. In (2.125) normalization is obtained by dividing the update term by the p-norm of the input vector, $\underline{x}(k)$. The regularization parameter, λ, is used to avoid excessively large updates in case of an occasionally small inputs. For $p = 2$, NLMP reduces to the Normalized-LMS algorithm [1].

The second algorithm, Normalized Least Mean Absolute Deviation (NLMAD), corresponds to the case of $p = 1$ in (2.125) with the following time update:

$$\underline{w}(n+1) = \underline{w}(n) + \beta \frac{\text{sign}[e(n)]}{||\underline{x}(n)||_1 + \lambda} \underline{x}(n). \tag{2.126}$$

This adaptation scheme is especially useful when the characteristic exponent, α, either is unknown or varying in time. Among the stable distributions the heaviest tail occur for the Cauchy distribution, $\alpha = 1$. By selecting $p = 1$ the update term is guaranteed to have a finite magnitude for all $1 < \alpha \leq 2$. Because of these reasons the NLMAD is a safe choice for the adaptation.

2.8.1 Use of Pre-nonlinearity in Adaptive Filtering

In this section the performance of the LMS and the RLS algorithms running on nonlinearly transformed data will be investigated. In this section, we consider the use of a softlimiter. The motivation behind this approach is to be able to reduce the effect of spiky characteristic of the α-stable data.This type of regularization have been used in robust signal processing applications [41]. It can be easily shown that any random process which is passed through a softlimiter has finite variance. Thus, the LMS and the RLS algorithms can be used in adaptation process after the input and reference signals have been soft-limited. The optimal filter coefficients which LMS and the RLS converge are biased. However, the bias so introduced can be kept at a reasonably small level by a proper selection of the threshold value in the softlimiter. The use of softlimiter reduces the spiky characteristics of input data hence a much smoother convergence can be expected. One noteworthy feature of this technique is that it has the same computational complexity as well-known LMS and RLS algorithms.

Because of the nonlinear mapping involved we call the proposed algorithms as the NMLMS and the NMRLS. A sample sequence of AR process disturbed by α-stable ($\alpha = 1.8$) noise and the output sequence after the softlimiter are shown in Figure 2.8.

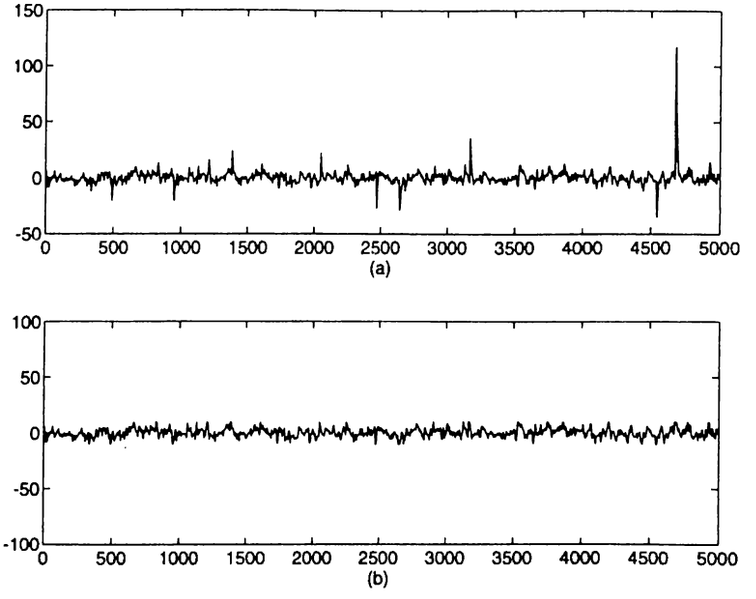

Figure 2.8: A sample AR process disturbed by α-stable ($\alpha = 1.8$) noise (a), and the output process after the soft limiter (b).

2.8.2 Computer Simulations

In simulation studies we consider $AR(N)$ α-stable processes, which are defined as follows,

$$x(n) = \sum_{i=1}^{N} a_i x(n-i) \; + \; u(n) \tag{2.127}$$

where $u(n)$ is a α-stable sequence of i.i.d random variables. The common distribution of $u(n)$ is chosen to be an even function ($\beta = 0$), and the gain factors are all set to one ($\gamma = 1$) without loss of generality. It can be shown that $x(n)$ will also be a α-stable random variable with the same characteristic exponent when $\{a_i\}$ is an absolutely summable sequence [40].

Two sets of simulation studies are performed. In the first set, the adaptation algorithms NLMAD, NLMP, LMAD, LMP and LMS are compared for a

second-order α-stable AR process with a fixed characteristic exponent, $\alpha = 1.2$. In the second set the performances of NLMAD, NLMP, NMLMS and NMRLS algorithms are compared for a second order α-stable AR process with different values of the characteristic exponent. For both sets, the tap weights are obtained by averaging 40 independent trials of the experiment and for each trial, a different computer realization of the process $\{u(n)\}$ is used. To get a fair comparison between algorithms the step sizes of adaptive algorithms are chosen in such a way that they all had a comparable steady-state variance. For both simulation set the coefficients of AR(2) is chosen as $a_1 = 0.99$ and $a_2 = -0.1$.

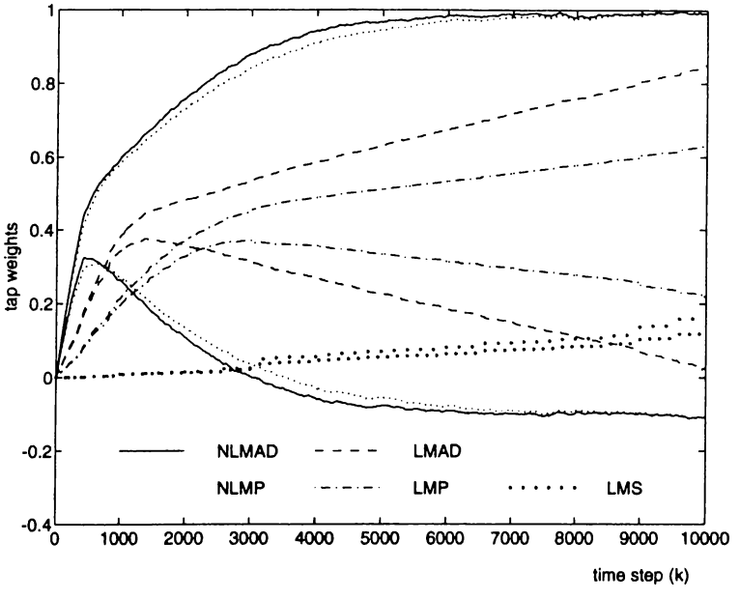

Figure 2.9: *Transient behavior of tap weights in the NLMAD, NLMP, LMAD, LMP and LMS algorithms with* $\alpha = 1.2$.

In the first part of the simulations, AR parameters $\underline{a}$ are estimated by a 2^{nd} order LMP, LMAD, NLMP, NLMAD and LMS algorithms. The plot of the tap weights is given in Figure 2.9. In the first part we observed that the normalized algorithms NLMAD and NLMP outperformed other algorithms. Therefore, in the second part the performances of NMLMS and NMRLS are only compared to NLMAD and NLMP algorithms.

In the second part of the simulations, AR parameters $\underline{a}$ are estimated by a 2^{nd} order NLMP, NLMAD, NMLMS and NMRLS algorithms for two different α-stable AR processes with $\alpha = 1.2$ and $\alpha = 1.8$. The plots of the tap weights

for NLMAD, NLMP and NMLMS algorithms are given in Figure 2.10 and Figure 2.11 for $\alpha = 1.2$ and $\alpha = 1.8$, respectively. The convergence performance of the tap weights for the NMRLS is given in Figure 2.12 for $\alpha = 1.2$ and $\alpha = 1.8$.

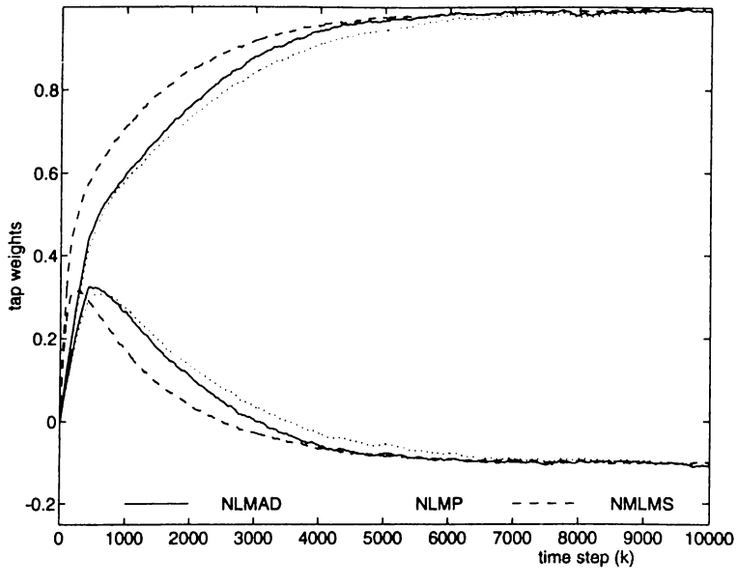

Figure 2.10: *Transient behavior of tap weights in the NMLMS, NLMAD, NLMP algorithms with* $\alpha = 1.2$.

2.9 Conclusion

In this chapter we present a new adaptive filtering algorithm which utilizes nonlinear transformation of both the input and the desired signals. The new algorithm has a number of useful features such as improved convergence speed, the capacity of reducing the spiky characteristic of the input data and operating under additive α-stable noise. Its major drawback is a bias introduced by the nonlinear transformation.

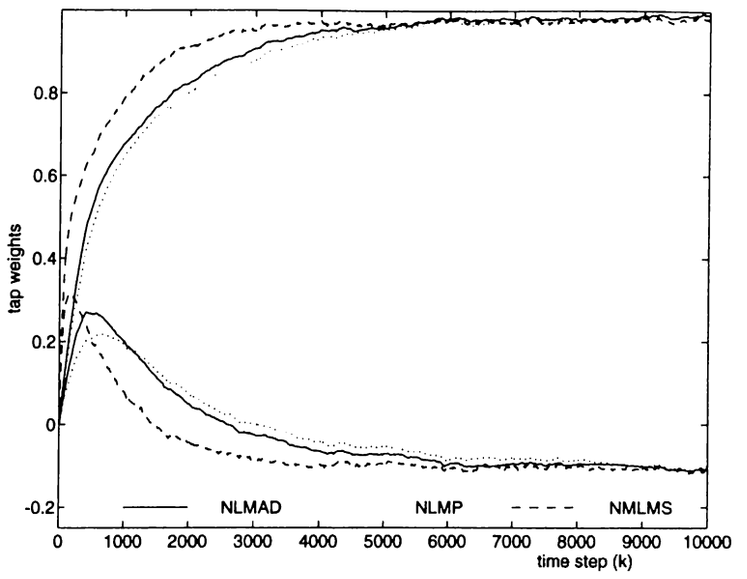

Figure 2.11: *Transient behavior of tap weights in the NMLMS, NLMAD, NLMP algorithms with* $\alpha = 1.8$.

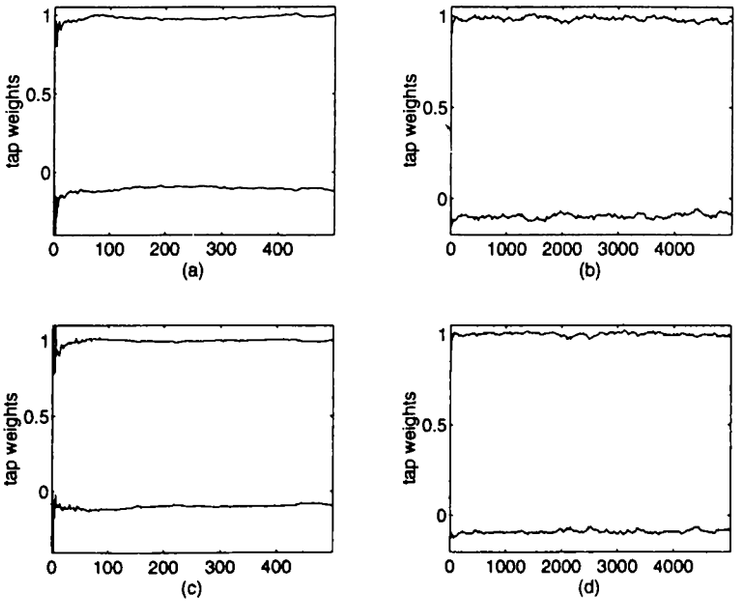

Figure 2.12: *Transient behavior of tap weights in the NMRLS algorithm with* $\alpha = 1.8$ *(a),(b), and* $\alpha = 1.2$ *(c),(d).*

Chapter 3

USE OF A PREWHITENING FILTER IN LMS ADAPTATION TO INCREASE CONVERGENCE SPEED

3.1 Introduction

The LMS algorithm is one of the most widely used algorithms for adaptive filtering due to its robustness and simplicity. Unfortunately, its convergence speed is highly dependent on the eigenvalue spread of the input autocorrelation matrix. There had been attempts in the past to improve the conditioning of the input correlation matrix by decomposing the input signal into a set of partially uncorrelated components via an orthogonal transform such as DFT or DCT. The main disadvantage of this type of preprocessing is the additional computational complexity introduced, and in the case of DFT, conversion of real data into imaginary data. It should also be noted that a fixed parameter

orthogonal transform may not give optimal results for all types of inputs signals.

It is known that the RLS algorithm achieves near optimum convergence rate by forming an estimate of R_x^{-1}, the inverse of the input autocorrelation matrix. This algorithm automatically adjusts the adaptive filter to whiten any input and also varies over time if the input is a nonstationary process.

In this chapter we present a new method to ameliorate convergence speed of the LMS adaptive filter. A linear filtering operation is performed on the input data and the desired signal to whiten the input signal while keeping the filter coefficients unaffected by the linear transformation. It is shown that the linear filtering of the input signal can decrease the condition number of the input correlation matrix without introducing additional computational complexity if the filter order is chosen small. The coefficients of the linear filter is obtained from a small order adaptive filter which is expected to work very fast.

3.2 Proposed Adaptive Algorithm

The proposed adaptive configuration is schematically represented in Fig. 3.1 . Here, $\underline{h} = [1 \ -\underline{h}_{M-1}^T]^T$ is an M^{th} order whitening filter whose coefficients are obtained in a least-square sense by solving the following minimization problem

$$\min_{\underline{h}_{M-1}} J(\underline{h}_{M-1}) = \sum_{n=1}^{L} \left(x(n) - \underline{h}_{M-1}^T \underline{x}_M(n-1)\right)^2 \tag{3.1}$$

where $\underline{x}_M(n-1)$ is the data vector formed by M most recent samples at time $n-1$

$$\underline{x}_M(n-1) = [x(n-1)\ x(n-2)\ \ldots x(n-M+1)]^T \tag{3.2}$$

Convolving the input and desired sequences by the filter $\underline{h}$ can be viewed as a linear prediction operation, where the current sample is predicted with a linear combination of M most recent samples of the input data. The error signal obtained as a result is fed to the adaptive filter which employs the LMS algorithm in the adaptation. Because of the linear filtering operation applied before adaptation, the proposed algorithm will be called as Linearly Filtered LMS (LFLMS).

Suppose that the relation between the input signal and the desired signal is given by $d(n) = \underline{f}^T \underline{x}(n)$. This means that the optimal Wiener solution for the tap weights converged by an Nth order adaptive filter is $\underline{f}$ itself. The question

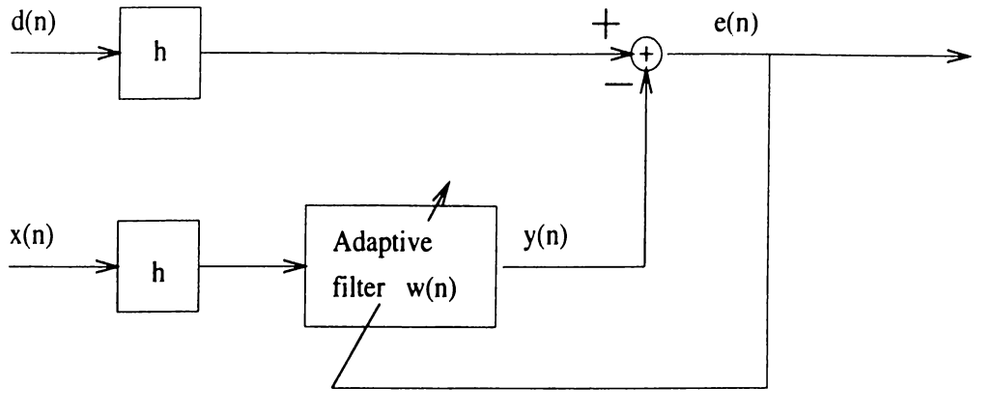

Figure 3.1: *Adaptive filtering block diagram*

here is whether the tap weights of the adaptive filter in the proposed system will converge to the same solution.

Suppose that the adaptation is performed by an N-tap adaptive filter and the order of the linear filter $\underline{h}$ is M. After the transformation we define the following signals

$$\hat{d}(n) = \underline{h}^T \underline{d}_M(n) \tag{3.3}$$

$$\hat{\underline{x}}(n) = \underline{h}^T \underline{x}_M(n) \tag{3.4}$$

where

$$\underline{d}_M(n) = [d(n)\ d(n-1) \ldots d(n-M+1)]^T \tag{3.5}$$

and $\underline{x}_M(n)$ is as defined in (3.2). The relation between $\hat{d}(n)$ and $\hat{\underline{x}}(n)$ can be found as follows

$$\begin{aligned}
\hat{d}(n) &= \underline{h}^T \underline{d}_M(n) = \underline{h}^T [d(n)\ \ d(n-1) \ldots d(n-M+1)]^T \\
&= \underline{h}^T [\underline{f}^T \underline{x}_N(n)\ \ \underline{f}^T \underline{x}_N(n-1) \ldots \underline{f}^T \underline{x}_N(n-M+1)]^T \\
&= \sum_{m=1}^{M} h_m \underline{f}^T \underline{x}_N(n-m+1) = \underline{f}^T \sum_{m=1}^{M} h_m \underline{x}_N(n-m+1) \\
&= \underline{f}^T [\underline{h}^T \underline{x}_M(n)\ \ \underline{h}^T \underline{x}_M(n-1) \ldots \underline{h}^T \underline{x}_M(n-N+1)]^T \\
&= \underline{f}^T \hat{\underline{x}}_N(n)
\end{aligned}$$

which is the same relation between $x(n)$ and $d(n)$. Therefore if the transformed input signal $\hat{x}(k)$ and the desired signal $\hat{d}(k)$ are fed to an adaptive filter, we expect that the adaptive algorithm converge to the same solution.

The key point here is that although there is no bias introduced, the spectrum of the input signal is flattened. If the order of the whitening filter were infinite, whitening operation would leave only the unpredictable part of $x(n)$ and the spectrum would be perfectly flat. In practice, we can not use an infinite order filter however, a few taps are sufficient to ameliorate the convergence properties of LMS algorithm. This is a direct result of the fact that the eigenvalue spread of the input autocorrelation matrix is bounded by the ratio of the maximum to the minimum of power spectrum.

3.2.1 Determination of Whitening Filter Coefficients

In Section 3.2 we have given a procedure for the determination of the filter coefficients. It is based on the minimization of the cost function given in (3.1). Optimizing filter coefficients are given by the solution to the normal equations [1]-[2]

$$\underline{h}_{M-1} = R_{M-1}^{-1}\underline{r}_{M-1} \tag{3.6}$$

where

$$R_{M-1} = E[\underline{x}_{M-1}(n)\underline{x}_{M-1}^{T}(n)] \tag{3.7}$$

is the input autocorrelation matrix at order $M-1$,

$$\underline{r}_{M-1} = [r(1)\ r(2)\ \ldots\ r(M)]^{T} \tag{3.8}$$

is the vector of cross correlations between $x(n)$ and $\underline{x}_{M-1}(n-1)$ and $r(m) = E[x(n)x(n-m)]$ is the autocorrelation sequence. Solution of (3.6) requires the computation of relevant statistics. If some a priori information is available about the signal statistics, we can simply derive optimal whitening filter coefficients via matrix inversion. A better way is to run an adaptive filter which takes $\hat{d}(n) = x(n)$ and $\hat{x}(n) = x(n-1)$ as its inputs prior to adaptation. In cases where the length of the adaptive filter is very long and the condition number of the input correlation matrix is very big, this approach can be very effective. However, it does not account for the changing statistics of inputs. In a nonstationary environment, one has to change the coefficients of the prewhitening filter in time to achieve proper operation. That is, the whitening filter coefficients must be adaptively determined as well. In the next section, we will describe this doubly adaptive configuration which will perform the desired operation.

In our simulations, we observed that the convergence properties of the LMS algorithm was insensitive to the small variations in the whitening filter coefficients. If the signal statistics are slowly varying about a certain value, satisfactory result can be obtained by using a constant prewhitening filter whose coefficients are obtained as described above.

3.3 Tracking In a Nonstationary Environment

We propose a slight modification to the adaptive filtering configuration described in section 3.2 by incorporating an adaptive section used to obtain prewhitening filter coefficients as shown in Fig 3.2 . The instantaneous error of this adaptive section, $e(n) = x(n) - \underline{h}_{M-1}^T(n)\underline{x}_{M-1}(n)$, is fed to the primary adaptive filter. The coefficients of the adaptive prewhitening filter is used to filter $d(n)$ also.

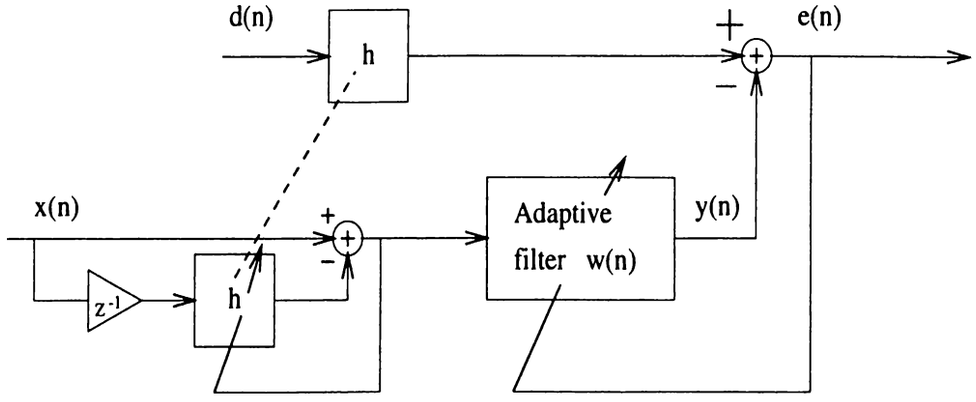

Figure 3.2: *Doubly adaptive LFLMS block diagram.*

In order to achieve a fast start-up, it is necessary to use a fast converging adaptive algorithm in the prewhitening section. We found RLS and Fast versions of RLS most convenient in our simulation studies. Variable step size LMS algorithms can also be used.

As the prewhitening filter converges from $\underline{h} = \underline{0}$ to their optimal values, the power of the inputs to the primary adaptive filter fluctuates widely and

therefore adaptation step must be made data adaptive to prevent loss of performance. Perhaps the most obvious modification to the LMS is to normalize the adaptation constant with respect to the power of the input signal. That is we replace the adaptation constant in LMS algorithm by

$$\hat{\mu} = \frac{\mu}{\underline{x}^T(n)\underline{x}(n)} \tag{3.9}$$

Incorporation of this simple normalization into the LMS update yields the so-called *Normalized* LMS algorithm whose update equation for the tap weights are given by [7]

$$\underline{w}(n+1) = \underline{w}(n) + \left\{ \frac{\mu}{\underline{x}^T(n)\underline{x}(n) + c_{\min}} \right\} e(n)\underline{x}(n) \tag{3.10}$$

where $c_{\min}$ is a small positive constant included to avoid division by zero. The Normalized LMS algorithm carries a computational overhead, though this can be limited to a single multiplication, addition and division at the expense of an increased storage requirement [7].

Note that the convolution by the prewhitening filter certainly changes the input signal and therefore the output $y(n) = \underline{w}^T(n)\underline{x}(n)$ is different from that of LMS. This is not a problem for system identification applications where the output of the filter is not needed. If the filter output is also needed, original input signal must be convolved by the impulse response of the primary adaptive filter.

3.4 Computational Cost

From the explanations above the proposed algorithm consists of three steps

1. Determination of prewhitening filter coefficients
2. Convolution of the input and desired signals with the impulse response of the prewhitening filter
3. Computation of adaptive filter parameters
4. Computation of the output of the adaptive filter.

If the prewhitening filter coefficients are determined a priori steps 2 and 3 requires a total of $2N + 2(M-1) + 1$ multiplications and 1 division. Step 4 requires extra N multiplications.

If the doubly adaptive configuration in Fig. 3.2 is used, steps 1 through 3 requires $2N+1+O(M^2)$ multiplications when the conventional RLS is used in the prewhitening filter. If we use Fast RLS or a variable step size LMS algorithm in adaptive prewhitening section, computational cost reduces to $2N + 1 + O(M)$ multiplications at each iteration.

Through a number of simulations, we observed that a fourth to tenth-order whitening filter significantly increases the rate of convergence. In applications where very long adaptive filters are required (such as echo cancelation where N is between 250 and 2000) the additional computational load introduced by the adaptive prewhitening section is negligible.

Finally, the number of multiplications required to update primary adaptive filter coefficients can be totally avoided by using *Ternary LMS* algorithm introduced in Chapter 4. In this case, only N multiplications are required to compute the output of the filter, if it is needed.

3.5 Computer Simulations

The proposed method was tested on a system identification experiment. The desired signal is derived by passing the input through a 7^{th}-order FIR filter whose coefficients were

$$\underline{h} = [0.9\ 0.8\ 0.7\ 0.6\ 0.5\ 0.4\ 0.3]^T \tag{3.11}$$

Computer generated white noise uncorrelated with the input signal is added to the output of the model system. The magnitude of the noise was set to give a power level of -60 dB relative to unity. Therefore at convergence, the minimum MSE achievable is -60 dB. The order of the adaptive filter was set to $N = 7$.

The input signal is generated by filtering Gaussian distributed white noise by a 32 tap linear phase FIR filter. The magnitude response of this noise coloring filter is shown in Fig. 3.3.a . According to the theoretical results, the convergence behavior of the LMS algorithm is determined by the input autocorrelation matrix. The condition number of the input autocorrelation matrix was approximately 2096.

The configuration in Fig. 3.1 was used to whiten the input data. Filter coefficients h_m were determined by a 7^{th} order one-step ahead linear prediction

filter. After convolving the input signal with the whitening filter, the condition number was reduced down to 60 to 300 resulting in a significant speed up of the convergence.

Fig. 3.3 shows the results of performing this system identification experiment as outlined above. 50 independent trials were averaged to obtain final results. The step sizes of the adaptive algorithms were adjusted to give the maximum convergence speed. Examination of Fig. 3.3 reveals that the linear transformation performed before the adaptation certainly ameliorated the performance of LMS algorithm, while keeping the optimal adaptive filter coefficients unchanged.

Fig. 3.4 shows the results of performing the same experiment with the doubly adaptive configuration shown in Fig. 3.2 . The adaptive algorithm used in prewhitening section was RLS and the normalized LMS algorithm used had a step size of 0.5. As observed, doubly adaptive LFLMS converged faster than both LMS and constant coefficient LFLMS. This is not a surprising result since doubly adaptive LFLMS determines prewhitening filter coefficients optimally for each data set. On the other hand, we used an approximate prewhitening filter for constant coefficient LFLMS and therefore this approach is suboptimal.

3.6 Conclusion

The results are summarized as follows.

- If there is a linear filtering relationship between the input and the desired signals, a linear transformation, if performed both on the input and desired signal, leave the coefficients of the adaptive filter unchanged.
- A small order whitening filter can be used to decorrelate the input signal. A direct consequence of this is an improvement in the convergence rate.
- Coefficients of the whitening filter can be adaptively determined allowing to adapt possible changes in the signal characteristics.
- Computational cost is increased by approximately $N + O(M)$ multiplications, though additional N multiplications can be avoided by using Ternary LMS algorithm.

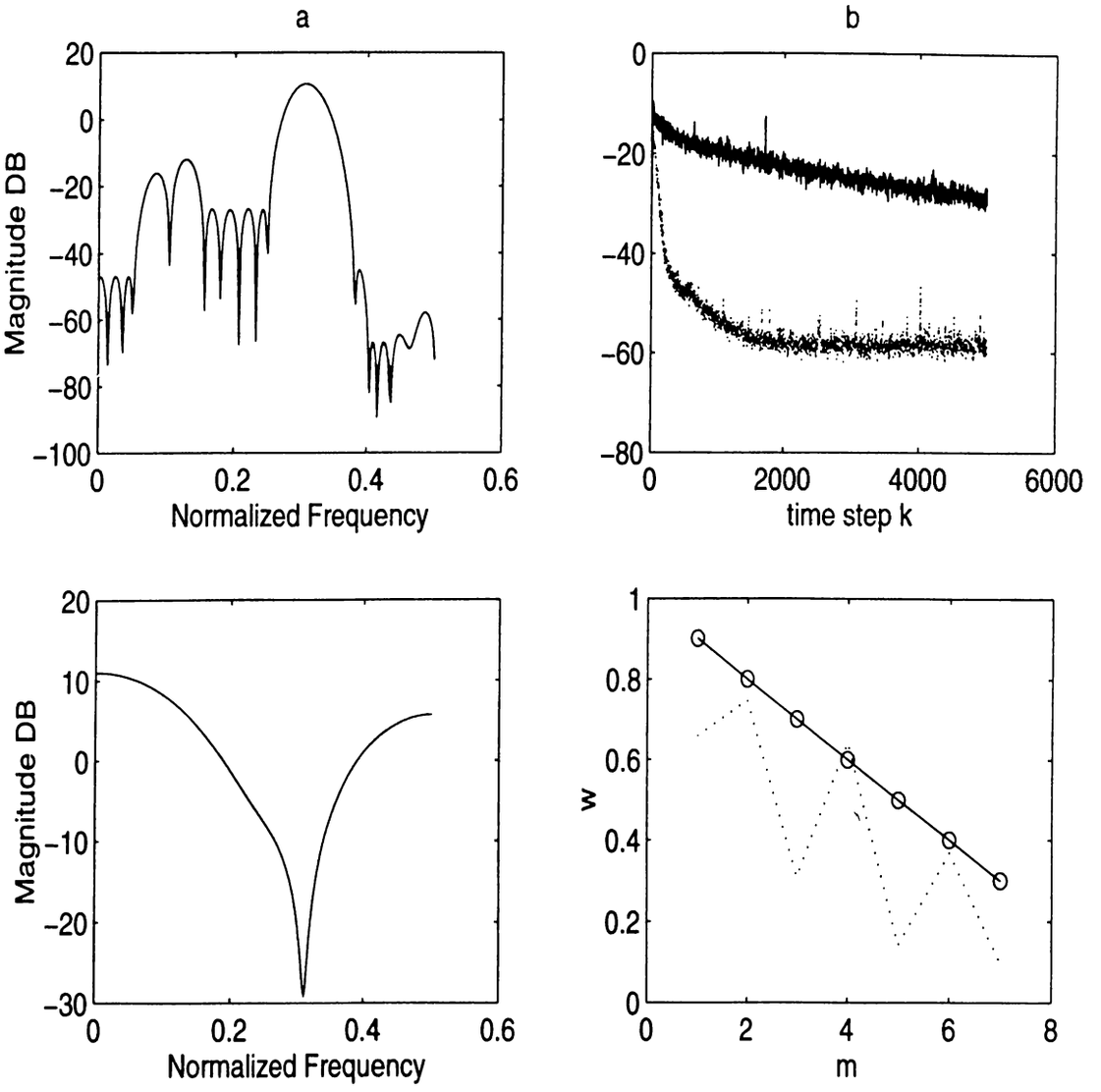

Figure 3.3: *Clockwise from upper left corner: Frequency Magnitude response of noise coloring filter, MSE for LMS(solid) and LFLMS(dot); final values of filter coefficients for LMS(dot) and LFLMS(solid)(optimal filter coefficients are indicated by circles); frequency magnitude response of prewhitening filter.*

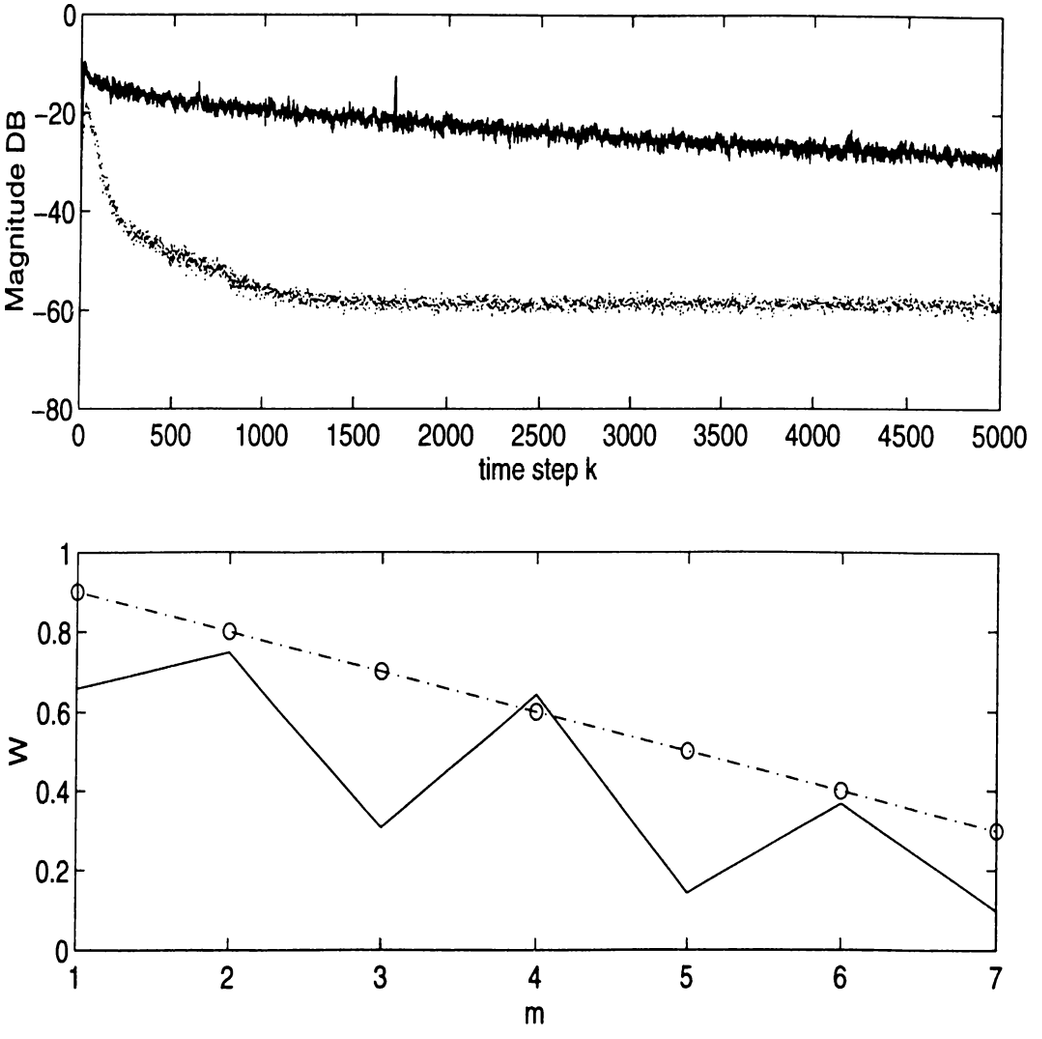

Figure 3.4: *MSE learning curves for LMS(solid) and doubly adaptive LFLMS(dotted); lower plot shows the final values of the coefficients for LMS(solid) and LFLMS(dashed).*

Chapter 4

QUANTIZED RECURSIVE LEAST SQUARES ALGORITHM

4.1 Introduction

Beginning with the very first applications of adaptive filters, much effort has been put into the search of different types of adaptation algorithms. The two most widely used adaptation algorithms, the Least Mean Squares (LMS) and Recursive Least Squares (RLS), have served as the benchmark for the new ones. Both LMS and RLS algorithms are designed to minimize the same type of cost function. It has been shown that when the signals are stationary, they converge to the ideal filter weights. However the convergence and the computational behaviors of these algorithms are different. Time updates for the LMS algorithm require $O(N)$ number of multiplications which is less than $O(N^2)$ multiplications required by the RLS algorithm. However, RLS converges faster, and, unlike LMS, convergence of RLS does not depend on the condition number of the input autocorrelation matrix. In order to decrease the number of multiplications in the time update of RLS algorithm from $O(N^2)$ to $O(N)$, various new update algorithms, that are known as fast RLS algorithms, have

been proposed. While still performing as accurate as RLS, the most efficient of these fast RLS algorithms reduces the number of multiplications to $7N$ [19-22].

In this chapter we will introduce a novel adaptation algorithm which is intended to further reduce the number of multiplications to update filter coefficients while preserving the characteristics of the RLS algorithm. In this new adaptation procedure roughly quantized auxiliary signals derived from the input and the desired signals are used. A very important property of these auxiliary signals is that they have exactly the same correlation functions as the input signals. We derive the new algorithm by defining a modified cost function of these auxiliary signals. It is shown that minimization of the modified cost function yields exactly the same estimate for the adaptive filter coefficients as that of the RLS algorithm. For amplitude limited signals only finitely many quantization levels are used. By using very few quantization levels, such as 9 or 16, a significant saving in the number of multiplications can be achieved. The trade-off between the variance of the estimated filter weights and the computational saving in the time updates is investigated.

4.2 Quantized Recursive Least Squares Algorithm

In this section we begin with a review of the minimum-mean-squared error adaptation, and provide the steps of ordinary Recursive Least Squares (RLS) algorithm. Then we set the framework of the new adaptation technique, and introduce the Quantized Recursive Least Squares (QRLS) algorithm. It will be shown that for amplitude-limited stationary signals, both QRLS and RLS converges to the same filter weights. The case of signals with no amplitude limitation will be discussed, and specifically for Gaussian signals, it will be shown that still very accurate results can be obtained by amplitude clipping the signals to three times their standard deviations. Following the analytical results, comparison of QRLS with RLS and LMS will be made based on simulation studies.

Given two input sequences $x(n)$ and $d(n)$, in minimum-mean-squared error adaptation, filter weights $\underline{w} = [w_1 \; w_2 \; \ldots \; w_N]^T$ are found to minimize the

expected value of the error squared

$$\min_{\underline{w}} \; J_{\underline{w}}(n) = E\left[(d(n) - \underline{x}^T(n)\underline{w})^2\right] \tag{4.1}$$

where $\underline{x}(n) = [x(n)\ x(n-1)\ \ldots\ x(n-N+1)]^T$ is the vector which contains N most recent samples of input signal. The optimal coefficient vector which minimizes the cost function in (4.1) is a solution to the normal equation

$$R_{xx}\hat{\underline{w}} = \underline{r}_{xd} \tag{4.2}$$

where $R_{xx} = E[\underline{x}(n)\underline{x}^T(n)]$ and $\underline{r}_{xd} = E[\underline{x}(n)d(n)]$.

In order to capture possibly time varying characteristics of the input signals, input autocorrelation matrix R_{xx} and the cross-correlations between input and desired signal $\underline{r}_{xd}$ should be estimated as time averages:

$$R_{xx}(n) = \sum_{k=1}^{n} \lambda^{n-k}\underline{x}(k)\underline{x}^T(k) \tag{4.3}$$

$$\underline{r}_{xd}(n) = \sum_{k=1}^{n} \lambda^{n-k}\underline{x}(k)d(k) \tag{4.4}$$

where $\lambda \in (0,\ 1]$ is an exponential forgetting factor. For nearly stationary input signals, a choice of λ close to 1 provides better statistics for the estimated correlations as time progress. However, it becomes more difficult for the adaptive filter to respond to rapid changes in the input signal characteristics. This well-known trade-off has been deeply investigated in the literature.

The recursive least squares algorithm (RLS) originated from a computationally efficient technique for inverting the input autocorrelation matrix arising in the normal equations. This is achieved by computing the inverse of the input autocorrelation matrix recursively by the help of the *matrix inversion lemma*[10]. The ordinary RLS algorithm is presented below:

$$\underline{k}(n) = \frac{G(n-1)\underline{x}(n)}{\lambda + \underline{x}^T G(n-1)\underline{x}(n)} \tag{4.5}$$

$$e(n) = d(n) - \underline{w}^T(n-1)\underline{x}(n) \tag{4.6}$$

$$\underline{w}(n) = \underline{w}(n-1) + \underline{k}(n)e(n) \tag{4.7}$$

$$G(n) = \lambda^{-1}\left[G(n-1) - \underline{k}(n)\underline{x}^T G(n-1)\right] \tag{4.8}$$

$$G(0) = \delta^{-1}I \tag{4.9}$$

$$\underline{w}(0) = \underline{0} \tag{4.10}$$

where $G(n) = R_{xx}^{-1}(n)$, I is the $n \times n$ identity matrix and δ is a small positive constant.

It is well known that [2] the parameter estimates will asymptotically converge to their true values provided that the equation error is a zero mean white noise and the underlying system is time invariant. As it is seen, the number of multiplications required to update the parameter estimates is $O(N^2)$. The RLS algorithm has many desirable features such as significant adaptation speed, low misadjustment and independence of convergence time from the input signal statistics.

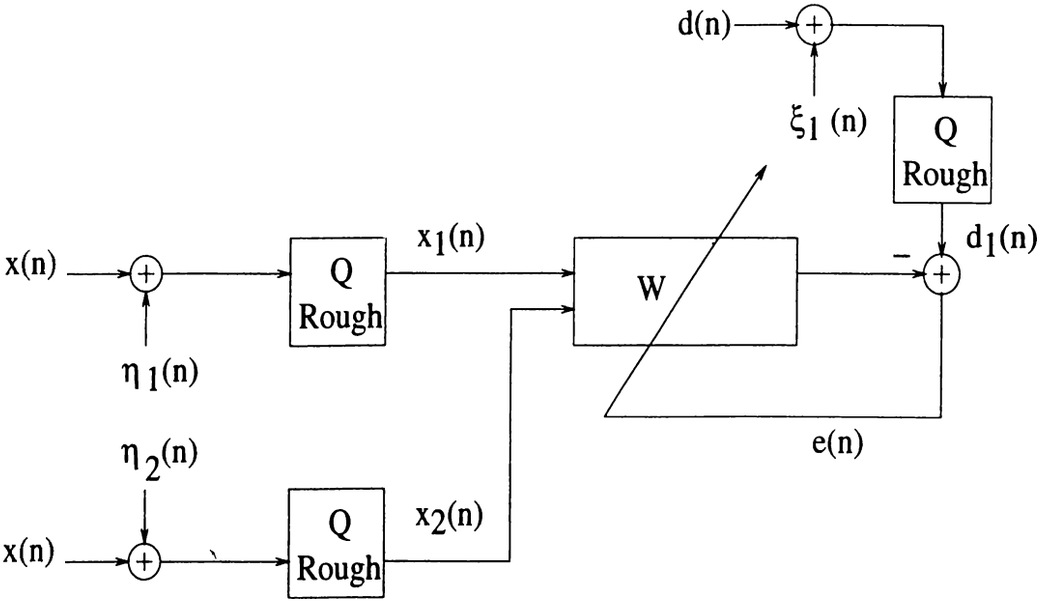

Figure 4.1: Quantized recursive least squares adaptive filter configuration

4.2.1 Derivation

In this section we will derive a recursive algorithm for solving the normal equations. The basic idea is to use *roughly quantized auxiliary signals* in place of input and the desired signals, which produce the same correlation functions. We introduce x_1 and x_2 in connection with x and d_1 in connection with d. These signals are obtained as in Fig. 4.1 by the following operations

$$x_1(n) = Q_1[x(n) + \eta_1(n)] \tag{4.11}$$

$$x_2(n) = Q_2[x(n) + \eta_2(n)] \tag{4.12}$$

$$d_1(n) = Q_3[d(n) + \xi_1(n)] \tag{4.13}$$

where Q_1, Q_2 and Q_3 are uniform quantizers and $\eta_1(n)$, $\eta_1(n)$ and $\xi_1(n)$ are zero mean, statistically independent random reference signals which are uniformly distributed within one quantization level of the quantizer of interest. In the Appendix, assuming that the input signals are bounded, it has been shown that the following equations are satisfied [12]

$$E[x_1(k)x_2(k+m)] = E[x(k)x(k+m)] \tag{4.14}$$

$$E[d_1(k)x_1(k+m)] = E[d(k)x(k+m)] \tag{4.15}$$

$$E[d_1(k)x_2(k+m)] = E[d(k)x(k+m)] \tag{4.16}$$

With the definitions above, we introduce a modified cost function of the following form

$$\hat{J}_w(n) = E\left[\left(d_1(n) - \underline{x}_1^T(n)\underline{w}(n)\right)\left(d_1(n) - \underline{x}_2^T(n)\underline{w}(n)\right)\right] \tag{4.17}$$

It can be easily shown that the optimal $\underline{w}$ which minimizes the cost function in (4.17) is a solution to

$$E[\underline{x}_1(n)\underline{x}_2^T(n) + \underline{x}_2(n)\underline{x}_1^T(n)]\underline{w}^* = E[(\underline{x}_1(n) + \underline{x}_2(n))d_1(n)] \tag{4.18}$$

Using the appropriate identities in (4.14)-(4.16) equation (4.18) can be put into the following form

$$R_{x_1x_2}\underline{w}^* = \underline{r}_{x_1d_1} \tag{4.19}$$

where $R_{x_1x_2} = E[\underline{x}_1(n)\underline{x}_2^T(n)]$ and $\underline{r}_{x_1d_1} = E[\underline{x}_1(n)d_1(n)]$. We have $R_{x_1x_2} = R_{xx}$ and $\underline{r}_{x_1d_1} = \underline{r}_{xd}$ by (4.14)-(4.16). Hence the optimal $\underline{w}^*$ is exactly the same as the $\underline{\hat{w}}$ in (4.2). In other words, the modified normal equation in 4.18, if it can be solved by some method, gives exactly the same estimate for the coefficient vector. The only difference is that, we have used roughly quantized signals to estimate the appropriate correlation functions.

In general, the input autocorrelation matrix $R_{x_1x_2}$ and the vector of cross-correlations between the input and the desired signals $\underline{r}_{x_1d_1}$ are not readily available in hand and must be estimated from finitely many samples of the input signals by some windowing operations. We assume input and the desired signals are ergodic and estimate $R_{x_1x_2}$ and $\underline{r}_{x_1d_1}$ by the following operations

$$R_{x_1x_2}(n) = \sum_{k=1}^{n} \lambda^{n-k}\underline{x}_1(k)\underline{x}_2^T(k) \tag{4.20}$$

$$\underline{r}_{x_1d_1}(n) = \sum_{k=1}^{n} \lambda^{n-k}\underline{x}_1(k)d_1(k). \tag{4.21}$$

where $\lambda \in (0,\ 1]$ is an exponential forgetting factor.

We can derive a recursive algorithm for the computation of $\underline{w}^*(n)$ based on the following update equations

$$R_{x_1x_2}(n) = \lambda R_{x_1x_2}(n-1) + \underline{x}_1(n)\underline{x}_2^T(n) \tag{4.22}$$

$$\underline{r}_{x_1d_1}(n) = \lambda \underline{r}_{x_1d_1}(n-1) + \underline{x}_1(n)d_1(n) \tag{4.23}$$

Assuming $R_{x_1x_2}(n)$ is nonsingular, we can apply the *matrix inversion lemma* [7] to the recursive equation (4.22)

$$R_{x_1x_2}^{-1}(n) = \lambda^{-1}R_{x_1x_2}^{-1}(n-1) - \frac{\lambda^{-2}R_{x_1x_2}^{-1}(n-1)\underline{x}_1(n)\underline{x}_2^T(n)R_{x_1x_2}^{-1}(n-1)}{1+\lambda^{-1}\underline{x}_2^T(n)R_{x_1x_2}^{-1}(n-1)\underline{x}_1(n)} \tag{4.24}$$

Defining

$$G(n) = R_{x_1x_2}^{-1}(n), \tag{4.25}$$

$$\underline{k}(n) = \frac{\lambda^{-1}G(n-1)\underline{x}_1(n)}{1+\lambda^{-1}\underline{x}_2^T(n)G(n-1)\underline{x}_1(n)} \tag{4.26}$$

we can rewrite (4.24) as

$$G(n) = \lambda^{-1}\left[G(n-1) - \underline{k}(n)\underline{x}_2^T(n)G(n-1)\right] \tag{4.27}$$

Multiplying (4.27) by $\underline{x}_1(n)$ and using the definition for $\underline{k}(n)$

$$\begin{aligned} G(n)\underline{x}_1(n) &= \lambda^{-1}G(n-1)\underline{x}_1(n) - \lambda^{-1}\underline{k}(n)\underline{x}_2^T(n)G(n-1)\underline{x}_1(n) \\ &= \underline{k}(n) + \lambda^{-1}\underline{k}(n)\underline{x}_2^T(n)G(n-1)\underline{x}_1(n) \\ &- \lambda^{-1}\underline{k}(n)\underline{x}_2^T(n)G(n-1)\underline{x}_1(n) \\ &= \underline{k}(n) \end{aligned} \tag{4.28}$$

Now, we can form a recursive update for $\underline{w}^*(n)$:

$$\begin{aligned} \underline{w}^*(n) &= G(n)\underline{r}_{x_1d_1}(n) \\ &= \lambda G(n)\underline{r}_{x_1d_1}(n-1) + G(n)\underline{x}_1(n)d_1(n) \\ &= \left[G(n-1) - \underline{k}(n)\underline{x}_2^T(n)G(n-1)\right]\underline{r}_{x_1d_1}(n-1) + G(n)\underline{x}_1(n)d_1(n) \\ &= \underline{w}^*(n-1) - \underline{k}(n)\underline{x}_2^T(n)\underline{w}^*(n-1) + G(n)\underline{x}_1(n)d_1(n) \\ &= \underline{w}^*(n-1) + \underline{k}(n)\left[d_1(n) - \underline{x}_2^T\underline{w}^*(n-1)\right] \end{aligned} \tag{4.29}$$

Equations (4.27) and (4.29) constitute the Quantized Recursive Least Squares algorithm (QRLS). It is initialized by choosing $G(0) = \delta^{-1}I$ and $\underline{w}^*(0) = \underline{0}$. It is seen that the steps of the QRLS algorithm are the same as those of the ordinary RLS algorithm with $d(n)$ replaced by $d_1(n)$ and $\underline{x}(n)$ replaced by $x_1(n)$ and $x_2(n)$.

Our development of QRLS algorithm is based on the assumption that the input signals are bounded in amplitude. We have shown that with this assumption, solutions to the original and quantized normal equations are exactly the same. While an input process coming from a p.d.f. of infinite extent, such as a Gaussian distribution, does not meet our assumption of inputs being bounded, by setting the maximum quantization level sufficiently high the occurrence of the process exceeding the maximum level can be made so rare that it does not affect the results significantly. A safe level that can be used for Gaussian distributed signals is three times the standard deviation, since this has a slight effect on the correlation function[12-13]. Detailed error analysis, summarized in the appendix, reveals that the performance of random reference correlator is not greatly different than a many bit correlator and as few as 8 (3-bits) or 16 (4-bits) levels in quantization provides good approximations [3]. This important property can be used to provide further significant savings in the number of floating point multiplications, leading to very efficient time updates.

We present a summary of QRLS in Table 4.1, including the number of operations required at each step assuming that all the quantizers use the same number of quantization levels.

4.2.2 Computational Cost

Representation of the input and the desired signals by 4, 3 or even 1-bit allows to compute a floating point multiplication by a few additions and shifts. Consider the multiplication of a floating point number by a 3-bit number. It is easy to see that the result of the multiplication can be calculated by at most 2 shifts and 2 additions. Thus, in the computation of the QRLS update parameters, multiplications by roughly quantized signals $x_1(n)$, $x_2(n)$ and $d_1(n)$ can be computed by a few shifts and a few additions. Adaptation performed on ternary quantized signals provides ultimate efficiency in terms of the total algebraic operations needed. Since a ternary quantized sample can only take three values, ± 1 or 0, in the worst case a simple sign change is enough to

Computation			
$n=0$: $G(0)=E_0$, $\underline{w}(0)=\underline{0}$ For each instant of time compute	MAD	Add.	ARCFM
$\underline{k}(n)=\frac{G(n-1)\underline{x}_1(n)}{\lambda+\underline{x}_2^T(n)G(n-1)\underline{x}_1(n)}$	1	N^2	N^2p+Np
$e(n)=d_1(n)-\underline{w}^{*T}(n-1)\underline{x}_2(n)$	0	N	Np
$\underline{w}^*(n)=\underline{w}^*(n-1)+\underline{k}(n)e(n)$	N	N	0
$G(n)=\lambda^{-1}\left[G(n)-\underline{k}(n)\left\{\underline{x}_2^T(n)G(n-1)\right\}\right]$	N^2+2N+1	$2N^2-1$	N^2p+N^2
$y(n)=\underline{w}^{*T}\underline{x}(n)$	N	$N-1$	0
Total	$3N+1$	$3N(N+1)-2$	$2N^2p+N^2+2Np$

Table 4.1: QRLS Algorithm. MAD = Multiplications and Divisions, ARCFM = Additions Required to Calculate Floating Point Multiplication. For 2-bits $p=1$, for 3-bits $p=2$.

compute the result of the multiplication.

In Table 1 at the fourth step, we assumed that the exponential forgetting factor was of the form $\lambda=\frac{2^l}{2^l+1}$ so that it takes only a shift and an addition to calculate the multiplication by λ.

The total number of floating point multiplications required to update filter parameters is still $O(N^2)$, although it has been reduced almost by a factor of four with respect to the conventional RLS algorithm.

4.2.3 Computer Simulation

QRLS algorithm was tested on a system identification experiment. In this simulation, input signal was zero mean, Gaussian distributed white noise. The desired signal was generated by passing the input signal through a 7^{th}-order FIR filter denoted as $\underline{h}(m)$, the model system.

Performance of the QRLS algorithm was compared to that of the conventional RLS algorithm. 4-bit, 3-bit and ternary quantizers were used. The power of the input and the desired signals were adjusted so that the maximum quantization level was three standard deviations above the mean. In order to show the possibility of using different number of quantization levels, we performed an experiment in which the desired signal was quantized into 4-bits, one of the inputs was quantized into 3-bits and the other was ternary quantized.

The QRLS and the RLS algorithms were initialized by $G(0) = \delta^{-1}I$. Here δ can be interpreted as a regularization constant or the initial signal energy. For the QRLS algorithm to work, δ must be chosen slightly larger than required by the RLS. As the number of quantization levels was decreased, we used larger and larger regularization constants. For 3 and 4 bit QRLS, satisfactory operation of the algorithm was achieved at $\delta = 0.1$. To obtain a smooth convergence curve we used $\delta = 0.5$ for 3-bit and 4-bit QRLS and for ordinary the RLS. For ternary quantized QRLS, we used $\delta = 7$, and for ternary, 3-bit, 4-bit QRLS we used $\delta = 3$.

Results of the simulations were plotted in Figs. 4.2-4.5, which were obtained by averaging 200 independent trials. As observed, the performance of 3-bit and 4-bit QRLS algorithms are very close to that of ordinary RLS. In fact they are hardly differentiable with respect to convergence speed. As the number of quantization levels decreased, the RLS algorithm outperforms the QRLS as expected.

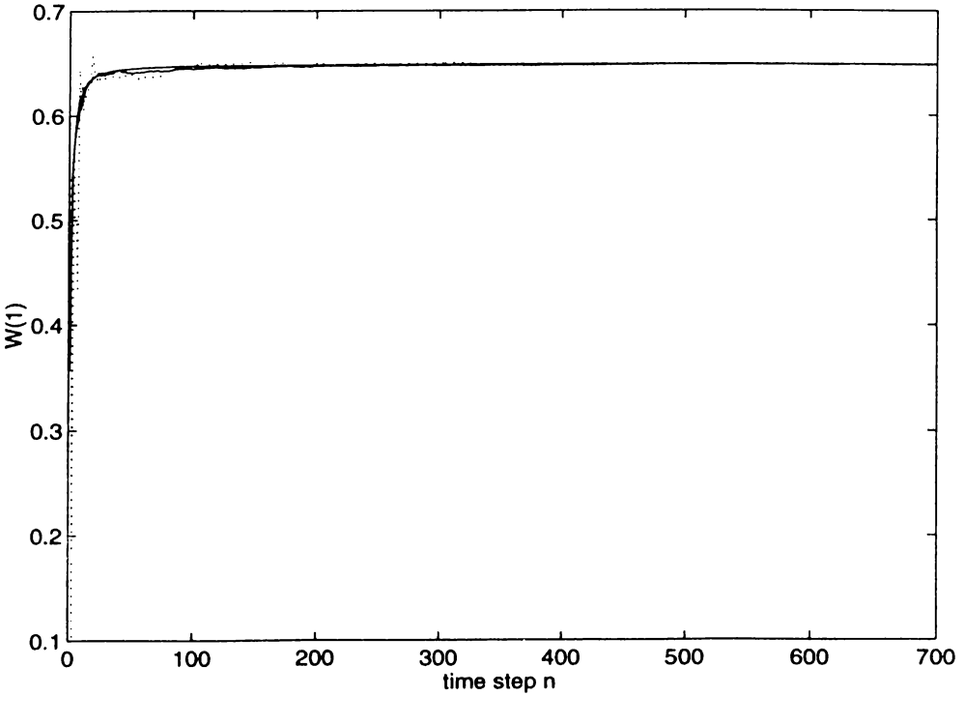

Figure 4.2: Convergence of RLS(solid), 4-bit QRLS(dash) and 3-bit QRLS(dot).

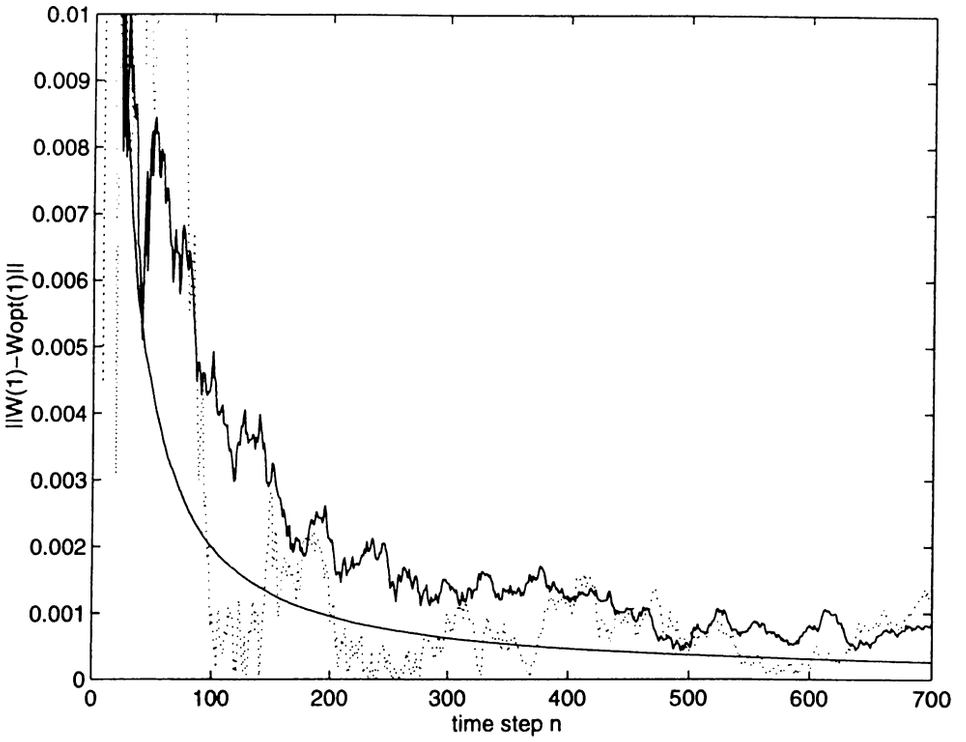

Figure 4.3: Coefficient deviations for RLS(solid), 4-bit QRLS(dash) and 3-bit QRLS(dot).

4.3 Fast Sequential Least Squares Adaptation Under Quantized Input and Desired Signals

In the previous section we have introduced the QRLS algorithm. The basic idea in developing the algorithm was the representation of the input and the desired signals by coarsely quantized signals. It is seen that the new algorithm yields substantial simplifications in calculating the multiplications involving coarsely quantized signals. This property can also be used to decrease the computational complexity of the fast RLS algorithms. In this section, we derive fast RLS algorithms, modified to run under roughly quantized signals.

In the RLS algorithm, the adaptation gain $\underline{k}(n)$ is used to update filter

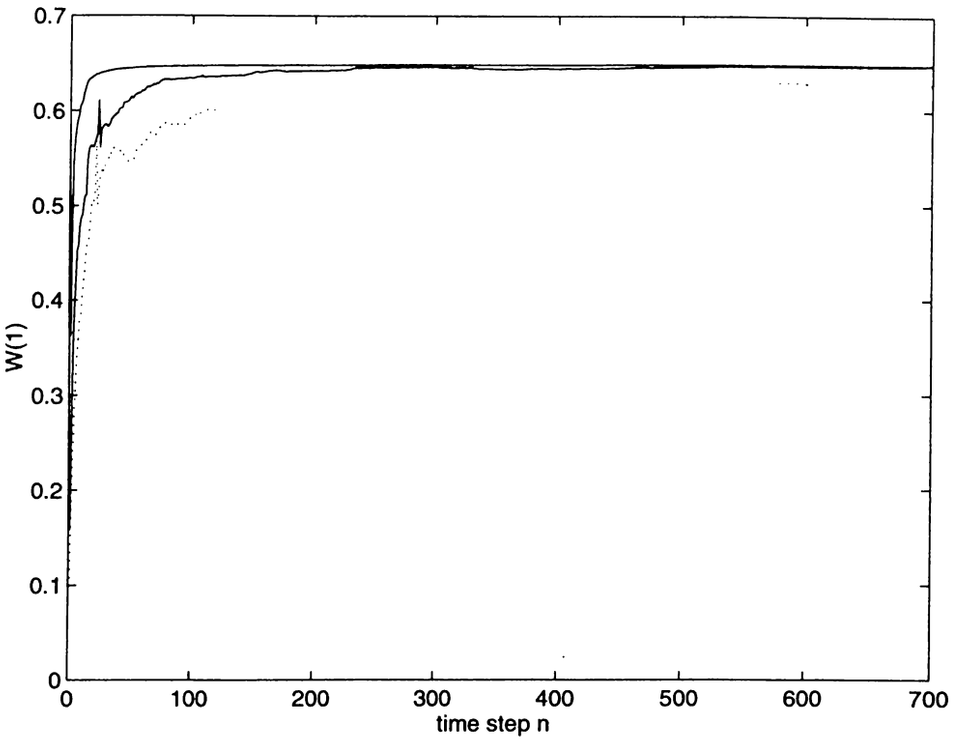

Figure 4.4: (a) Convergence of the RLS(solid), ternary(x_1), 3-bit(x_2) 4-bit(d_1) QRLS(dash) and ternary QRLS(dot).

coefficients which is itself updated by the help of the inverse input autocorrelation matrix. In fast algorithms, forward and backward prediction parameters are used to update the desired gain vector.

We define the forward and the backward least squares linear prediction cost functions as the following respectively:

$$E_a(n) = E\left[\left(x_1(n) - \underline{a}^T(n)\underline{x}_1(n)\right)\left(x_2(n) - \underline{a}^T(n)\underline{x}_2(n)\right)\right] \quad (4.30)$$

$$E_b(n) = E\left[\left(x_1(n-N) - \underline{b}^T(n)x_1(n)\right)\left(x_2(n-N) - \underline{b}^T(n)\underline{x}_2(n)\right)\right] \quad (4.31)$$

Note that the above definition for the forward and the backward least squares cost functions produces exactly the same forward and backward prediction errors as in the case of finely quantized signals. Minimization of the cost functions in (4.30)-(4.31) leads to the optimal forward and backward coefficient vectors given by

$$\underline{a}(n) = G(n-1)\underline{r}_N^a(n) \quad (4.32)$$

$$\underline{b}(n) = G(n)\underline{r}_N^b(n) \quad (4.33)$$

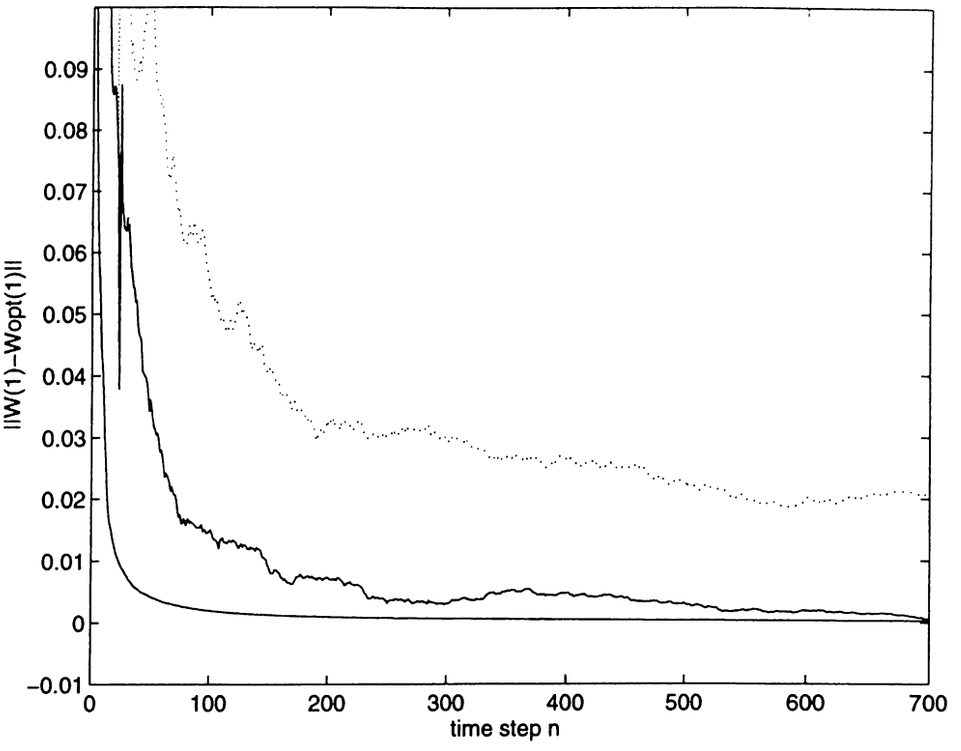

Figure 4.5: Coefficient deviations for RLS(solid), ternary(x_1), 3-bit(x_2) 4-bit(d_1) QRLS(dash) and ternary QRLS(dot).

where

$$\underline{r}_N^a(n) = E\left[x_1(n)\underline{x}_2(n)\right] \tag{4.34}$$

$$\underline{r}_N^b(n) = E\left[x_1(n-N)\underline{x}_2(n)\right] \tag{4.35}$$

and $G(n)$ is the inverse of the input autocorrelation matrix as defined in the previous section. The coefficients of the forward and the backward predictors are updated by

$$\underline{a}(n+1) = \underline{a}(n) + \underline{k}(n)e_a(n+1) \tag{4.36}$$

$$\underline{b}(n+1) = \underline{b}(n) + \underline{k}(n+1)e_b(n+1) \ . \tag{4.37}$$

where

$$e_a(n+1) = x_2(n+1) - \underline{a}^T(n)\underline{x}_2(n) \tag{4.38}$$

$$e_b(n+1) = x_2(n+1) - \underline{b}^T(n)\underline{x}_2(n) \ . \tag{4.39}$$

These are *a priori* prediction errors. The updated coefficients are used to calculate *a posteriori* prediction errors:

$$\epsilon_a(n+1) = x_1(n+1) - \underline{a}^T(n)\underline{x}_1(n) \quad (4.40)$$

$$\epsilon_b(n+1) = x_2(n+1) - \underline{b}^T(n)\underline{x}_1(n) \quad (4.41)$$

In a practical implementation expected values should be replaced by the time averages. In this case, the prediction error energies are estimated by

$$E_a(n) = \sum_{k=1}^{n} \lambda^{n-k} \left(x_1(n) - \underline{a}^T(n)\underline{x}_1(n)\right)\left(x_2(n) - \underline{a}^T(n)\underline{x}_2(n)\right) \quad (4.42)$$

$$E_b(n) = \sum_{k=1}^{n} \left(x_1(n-N) - \underline{b}^T(n)x_1(n)\right)\left(x_2(n-N) - \underline{b}^T(n)\underline{x}_2(n)\right) \quad (4.43)$$

Similarly, $\underline{r}_N^a(n)$ and $\underline{r}_N^b(n)$ are estimated by

$$\underline{r}_N^a(n) = \sum_{k=1}^{n} \lambda^{n-k} x_1(n)\underline{x}_2(n) \quad (4.44)$$

$$\underline{r}_N^b(n) = \sum_{k=1}^{n} \lambda^{n-k} x_1(n-N)\underline{x}_2(n) \quad . \quad (4.45)$$

By using the recurrence relations for $\underline{a}(n+1)$ and $\underline{b}(n+1)$ and $\underline{r}_N^a(n+1)$ and $\underline{r}_N^b(n+1)$ it can be easily shown that the prediction error energies can be recursively calculated by the following equations

$$E_a(n+1) = \lambda E_a(n) + e_a(n+1)\epsilon_a(n+1) \quad (4.46)$$

$$E_b(n+1) = \lambda E_b(n) + e_b(n+1)\epsilon_b(n+1) \quad . \quad (4.47)$$

These are the fundamental equations which are used in the fast RLS algorithms.

4.3.1 Fast Algorithm Based On A Priori Errors:

We consider $(N+1 \times N+1)$ input autocorrelation matrix denoted by $R_{x_1x_2}^{N+1}(n+1)$. It can be partitioned in two ways

$$R_{x_1x_2}^{N+1}(n+1) = \begin{bmatrix} \sum_{k=1}^{n+1} \lambda^{n-k+1} x_1(k)x_2(k) & [\underline{r}_N^a(n+1)]^T \\ \underline{r}_N^a(n+1) & R_{x_1x_2}(n) \end{bmatrix} \quad (4.48)$$

and

$$R_{x_1x_2}^{N+1}(n+1) = \begin{bmatrix} R_{x_1x_2}(n+1) & [\underline{r}_N^b(n+1)]^T \\ \underline{r}_N^b(n+1) & \sum_{k=1}^{n+1} \lambda^{n-k+1} x_1(k-N)x_2(k-N) \end{bmatrix} \quad . \quad (4.49)$$

Our objective is to find a recursive update for the gain vector defined as

$$R_{x_1x_2}(n+1)\underline{k}(n+1) = \underline{x}_1(n+1) \quad . \quad (4.50)$$

Using the partitioning in (4.48) we calculate the following

$$R_{x_1x_2}^{N+1}(n+1) \begin{bmatrix} 0 \\ \underline{k}(n) \end{bmatrix} = \begin{bmatrix} [\underline{r}_N^a(n+1)]^T \underline{k}(n) \\ \underline{x}_1(n) \end{bmatrix} = \begin{bmatrix} \underline{a}^T(n+1)\underline{x}_1(n) \\ \underline{x}_1(n) \end{bmatrix} \quad (4.51)$$

Introducing the a posteriori prediction error, we can put the above equation into the following form

$$R_{x_1x_2}^{N+1}(n+1) \begin{bmatrix} 0 \\ \underline{k}(n) \end{bmatrix} = \underline{x}_1^{N+1}(n+1) - \begin{bmatrix} \epsilon_a(n+1) \\ 0 \end{bmatrix} \quad (4.52)$$

where $\underline{x}_1^{N+1}(n+1)$ is the vector which contains $N+1$ most recent samples of the input data at time $(n+1)$. By a similar way, it can be easily shown that the following equation is satisfied

$$R_{x_1x_2}^{N+1}(n+1) \begin{bmatrix} \underline{k}(n+1) \\ 0 \end{bmatrix} = \underline{x}_1^{N+1}(n+1) - \begin{bmatrix} 0 \\ \epsilon_b(n+1) \end{bmatrix} \quad (4.53)$$

The adaptation gain at dimension $N+1$, denoted by $\underline{k}^{N+1}(n+1)$ with the above notation is defined by

$$R_{x_1x_2}^{N+1}(n+1)\underline{k}^{N+1}(n+1) = \underline{x}_1^{N+1}(n+1) \quad (4.54)$$

Then, (4.52) can be written as

$$R_{x_1x_2}^{N+1}(n+1) \left[\underline{k}^{N+1}(n+1) - \begin{bmatrix} 0 \\ \underline{k}(n) \end{bmatrix} \right] = \begin{bmatrix} \epsilon_a(n+1) \\ 0 \end{bmatrix} \quad (4.55)$$

Similarly, (4.53) can be written as

$$R_{x_1x_2}^{N+1}(n+1)\left[\underline{k}^{N+1}(n+1)-\begin{bmatrix}\underline{k}(n+1)\\0\end{bmatrix}\right]=\begin{bmatrix}0\\\epsilon_b(n+1)\end{bmatrix} \tag{4.56}$$

The linear prediction matrix equations will be used to compute $\underline{k}^{N+1}(n+1)$ from $\underline{k}(n)$, and then $\underline{k}(n+1)$ from $\underline{k}^{N+1}(n+1)$. The forward linear prediction matrix equation, combining (4.32) and (4.38) is

$$R_{x_1x_2}^{N+1}(n+1)\begin{bmatrix}1\\-\underline{a}(n+1)\end{bmatrix}=\begin{bmatrix}E_a(n+1)\\\underline{0}\end{bmatrix} \tag{4.57}$$

By identifying the factors in (4.55) and (4.57) yields, the following recursion:

$$\underline{k}^{N+1}(n+1)=\begin{bmatrix}0\\\underline{k}(n)\end{bmatrix}+\frac{\epsilon_a(n+1)}{E_a(n+1)}\begin{bmatrix}1\\-\underline{a}(n+1)\end{bmatrix} \tag{4.58}$$

The backward linear prediction matrix equation is

$$R_{x_1x_2}^{N+1}(n+1)\begin{bmatrix}-\underline{b}(n+1)\\1\end{bmatrix}=\begin{bmatrix}\underline{0}\\E_b(n+1)\end{bmatrix} \tag{4.59}$$

Similarly, identifying the factors in (4.56) and (4.59) yields

$$\underline{k}^{N+1}(n+1)=\begin{bmatrix}\underline{k}(n+1)\\0\end{bmatrix}+\frac{\epsilon_b(n+1)}{E_b(n+1)}\begin{bmatrix}-\underline{b}(n+1)\\1\end{bmatrix} \tag{4.60}$$

Since $\underline{k}(n)$ is available, (4.55) can be used to compute $\underline{k}^{N+1}(n+1)$ and then $\underline{k}(n+1)$ can be computed by (4.60). We do not need to calculate the scalar factor appearing in (4.60) since it is already available. The partitioning of $\underline{k}^{N+1}(n+1)$ as

$$\underline{k}^{N+1}(n+1)=\begin{bmatrix}\underline{M}(n+1)\\m(n+1)\end{bmatrix} \tag{4.61}$$

is useful in the gain update. The scalar factor in (4.60) is identified as

$$m(n+1)=\frac{\epsilon_b(n+1)}{E_b(n+1)} \tag{4.62}$$

giving the following adaptation gain update

$$\underline{k}(n+1) = \underline{M}(n+1) + m(n+1)\underline{b}(n+1) \tag{4.63}$$

To update the adaptation gain, we need the updated backward prediction coefficients $\underline{b}(n+1)$. Substitution of (4.37) into (4.63) gives an expression for the adaptation gain in terms of the known quantities:

$$\underline{k}(n+1) = \frac{1}{1 - m(n+1)eb(n+1)}[\underline{M}(n+1) + \underline{b}(n)m(n+1)] \tag{4.64}$$

Overall, we have derived a fast algorithm based on a priori errors which used coarsely quantized signals as its inputs. The LS initialization is obtained by taking $\underline{a}(n) = \underline{b}(n) = \underline{k}(n) = 0$, and $E_a(0) = E_0$, a small positive constant. This algorithm is sometimes called as fast Kalman algorithm. Examination of the derivation of the algorithm reveals that it is the same as the fast Kalman algorithm except for the definition of the forward and backward prediction cost functions. With the definitions in (4.30) and (4.31), any fast algorithm can be adapted to run under coarsely quantized signals. The computational organization of the algorithm is given in Table. 4.2, together with the count of operations involved.

4.3.2 Computational Cost

Original form of the fast Kalman algorithm requires $8N + 4$ multiplications and 2 divisions for the adaptation gain updating, and $2N$ multiplications for the filtering section. We could reduce the number of multiplications down to $5N + 4$ for the adaptation gain updating and N for the filtering, saving $3N$ multiplications in total. Examination of the computational organization of fast Kalman algorithm reveals that, floating point multiplications that must be carried out to calculate inner products by the input data vectors $\underline{x}_1(n)$ and $\underline{x}_2(n)$ are replaced by simple additions, since these vectors consists of roughly quantized signals. In general, we expect that any adaptive algorithm running on roughly quantized inputs would result in computational savings whenever multiplications with roughly quantized signals are performed. There are a number of different fast RLS algorithms, and they can be easily modified to adopt the structure depicted in Fig. (4.1) resulting in a more computationally efficient algorithm with performance comparable to that of the unmodified algorithm.

Computation			
$n=0$: $E_a(0)=E_0 \quad \underline{w}(0)=\underline{a}(0)=\underline{b}(0)=\underline{0}$ For each instant of time compute	MAD	Add.	ARC.
Adaptation Gain updating			
$e_a(n+1)=x_2(N+1)-\underline{a}^T(n)\underline{x}_2(n)$	0	N	Np
$\underline{a}(n+1)=\underline{a}(n)+\underline{k}(n)e_a(n+1)$	N	N	0
$\epsilon_a(n+1)=x_1(N+1)-\underline{a}^T(n+1)\underline{x}_1(n)$	0	N	Np
$E_a(n+1)=\lambda E_a(n)+e_a(n+1)\epsilon_a(n+1)$	2	1	0
$\begin{bmatrix}\underline{M}(n+1)\\ m(n+1)\end{bmatrix}=\begin{bmatrix}0\\ \underline{k}(n)\end{bmatrix}+\frac{e_a(n+1)}{E_a(n+1)}\begin{bmatrix}1\\ -\underline{a}(n+1)\end{bmatrix}$	$N+2$	$N+1$	0
$e_b(n+1)=x_2(n+1)-\underline{b}^T(n)\underline{x}_2(n)$	0	N	Np
$\underline{k}(n+1)=\frac{\underline{M}(n+1)+\underline{b}(n)m(n+1)}{1-m(n+1)e_b(n+1)}$	$2N+2$	$N+1$	0
$\underline{b}(n+1)=\underline{b}(n)+\underline{k}(n+1)e_b(n+1)$	N	N	0
Adaptive filter			
$e(n+1)=d_1(n+1)-\underline{w}^T(n)\underline{x}_2(n)$	0	N	Np
$\underline{w}(n+1)=\underline{w}(n)+\underline{k}(n+1)e(n+1)$	N	N	0
Total	$6N+6$	$7N+3$	$4Np$

Table 4.2: Computational organization of the fast RLS algorithm based on a priori prediction errors

Here, we give two different fast RLS algorithms which are modified to accept roughly quantized signals as their inputs in Table (4.3) and (4.4).

4.4 Least Mean Square Adaptation Under Roughly Quantized Input and Desired Signals

In the preceding sections we have shown that the RLS and the fast RLS adaptive filters could be run under coarsely quantized signals. It is also possible to use the LMS adaptation algorithm with coarsely quantized signals.

Computation			
$n=0$: $E_a(0)=E_0 \quad \underline{w}(0)=\underline{a}(0)=\underline{b}(0)=\underline{0}$ For each instant of time compute	MAD	Add.	ARC.
Adaptation Gain updating			
$e_a(n+1)=x_2(N+1)-\underline{a}^T(n)\underline{x}_2(n)$	0	N	Np
$\underline{a}(n+1)=\underline{a}(n)+\frac{\underline{k}(n)e_a(n+1)}{\alpha(n)}$	$N+1$	N	0
$E_a(n+1)=\lambda\left(E_a(n)+\frac{e_a(n+1)\epsilon_a(n+1)}{\alpha(n)}\right)$	3	1	0
$\begin{bmatrix}\underline{M}(n+1)\\ m(n+1)\end{bmatrix}=\begin{bmatrix}0\\ \underline{k}(n)\end{bmatrix}+\frac{e_a(n+1)}{E_a(n+1)}\begin{bmatrix}1\\ -\underline{a}(n+1)\end{bmatrix}$	$N+2$	$N+1$	0
$e_b(n+1)=x_2(n+1)-\underline{b}^T(n)\underline{x}_2(n)$	0	N	Np
$\underline{k}(n+1)=\underline{M}(n+1)+\underline{b}(n)m(n+1)$	N	N	0
$\alpha_1(n+1)=\alpha(n)+\frac{e_a^2(n+1)}{E_a(n)}$	2	1	0
$\alpha(n+1)=\alpha_1(n+1)-m(n+1)e_b(n+1)$	1	1	0
$E_b(n+1)=\lambda\left(E_b(n)+\frac{e_b^2(n+1)}{\alpha(n+1)}\right)$	3	1	0
$\underline{b}(n+1)=\underline{b}(n)+\frac{\underline{k}(n+1)e_b(n+1)}{\alpha(n+1)}$	$N+1$	N	0
Adaptive filter			
$e(n+1)=d_1(n+1)-\underline{w}^T(n)\underline{x}_2(n)$	0	N	Np
$\underline{w}(n+1)=\underline{w}(n)+\underline{k}(n+1)e(n+1)$	N	N	0
Total	$5N+13$	$8N+5$	$3Np$

Table 4.3: Computational organization of the fast RLS algorithm based on all prediction errors.

Computation			
$n = 0$: $E_a(0) = E_0 \quad \underline{w}(0) = \underline{a}(0) = \underline{b}(0) = \underline{0}$ For each instant of time compute	MAD	Add.	ARC.
Adaptation Gain updating			
$e_a(n+1) = x_2(N+1) - \underline{a}^T(n)\underline{x}_2(n)$	0	N	Np
$\underline{a}(n+1) = \underline{a}(n) + \frac{\underline{k}(n)e_a(n+1)}{\alpha(n)}$	$N+1$	N	0
$E_a(n+1) = \lambda\left(E_a(n) + \frac{e_a(n+1)e_a(n+1)}{\alpha(n)}\right)$	3	1	0
$\begin{bmatrix} \underline{M}(n+1) \\ m(n+1) \end{bmatrix} = \begin{bmatrix} 0 \\ \underline{k}(n) \end{bmatrix} + \frac{e_a(n+1)}{E_a(n+1)} \begin{bmatrix} 1 \\ -\underline{a}(n+1) \end{bmatrix}$	$N+2$	$N+1$	0
$\underline{k}(n+1) = \underline{M}(n+1) + J_N m(n+1)\underline{a}(n)$	$N+1$	N	0
$\alpha(n+1) = \lambda + \underline{k}^T(n+1)\underline{x}_1(n+1)$	0	N	Np
Adaptive filter			
$e(n+1) = d_1(n+1) - \underline{w}^T(n)\underline{x}_2(n)$	0	N	Np
$\underline{w}(n+1) = \underline{w}(n) + \underline{k}(n+1)e(n+1)$	N	N	0
Total	$4N+7$	$7N+2$	$2Np$

Table 4.4: Computational organization of the simplified fast RLS algorithm

The conventional LMS algorithm has the following update for the tap weights:

$$\underline{w}(n+1) = \underline{w}(n) + \mu e(n)\underline{x}(n) \tag{4.65}$$

where

$$e(n) = d(n) - \underline{w}^T(n)\underline{x}(n) \quad . \tag{4.66}$$

Under the generally accepted assumption that the tap weights and input signal are independent of each other at convergence, it can be easily shown that the tap weights converge to

$$\underline{w}_\infty = R_{xx}^{-1}\underline{r}_{xd} \quad . \tag{4.67}$$

The LMS algorithm requires $2N+1$ multiplications and $2N$ additions to update the filter coefficients. Because of its computational and conceptual simplicity, it has been the most popular adaptation algorithm for the last two decades.

We propose the following modification in the update equation by introducing coarsely quantized signals $x_1(n)$, $x_2(n)$ and $d_1(n)$:

$$\underline{w}(n+1) = \underline{w}(n) + \mu\hat{e}(n)\underline{x}_1(n) \tag{4.68}$$

where

$$\hat{\epsilon}(n) = d_1(n) - \underline{w}^T(n)\underline{x}_2(n) \quad . \tag{4.69}$$

Filter output is computed by

$$y(n) = \underline{w}^T(n)\underline{x}(n) \quad , \tag{4.70}$$

It can be easily shown that, filter coefficients converge to

$$\underline{w}_\infty = R^{-1}_{x_1x_2}\underline{r}_{x_1d_1} \tag{4.71}$$

Therefore the quantized LMS (QLMS) algorithm gives an unbiased estimate for the filter coefficients, since $R_{x_1x_2} = R_{xx}$ and $\underline{r}_{x_1d_1} = \underline{r}_{xd}$.

Ternary quantized LMS algorithm is an alternative to Clipped LMS, or signed regressor algorithm whose update equation for the tap weights is given by

$$\underline{w}(n+1) = \underline{w}(n) + \mu e(n)\text{sign}[\underline{x}(n)] \quad . \tag{4.72}$$

Although it has been shown that the clipped LMS algorithm converges in the mean to the optimal Wiener solution for white Gaussian inputs, simple input sequences can be found which produce stable results for LMS but unstable for clipped LMS. Another problem is that due to the crude approximation to the gradient, considerable slow-down in the convergence is manifested[2]-[7]. Ternary quantized LMS algorithm does not suffer from these problems, since it converges in the mean to the same solution as LMS. Simulations have shown that convergence speed is comparable to the LMS, although increased steady-state MSE may be observed due to higher error variance of random reference correlator.

4.4.1 Computational Cost

For system identification applications which require only the computation of model system parameters, the quantized LMS algorithm requires no multiplications. Ultimate efficiency in computational cost is achieved if the input and the desired signals are ternary quantized. In this case, no additions are required to calculate the result of the multiplication by a ternary quantized signal sample, a simple sign change is sufficient. For the applications which require the output of the filter as well, we need to compute additional N multiplications and N additions.

4.4.2 Computer Simulation

Quantized LMS algorithm was tested on a one-step-ahead prediction experiment and the results were compared to that of the LMS. In this simulation, input signal was a second-order AR(2) process

$$x(n) = 0.975x(n-1) - 0.95x(n-2) + u(n) \tag{4.73}$$

Here, $u(n)$ was a white Gaussian process whose variance was adjusted to give unity steady state variance: $\sigma_x^2 = 1$. the desired signal was

$$d(n) = x(n+1) \quad . \tag{4.74}$$

Ideally, adaptive filter weights must converge to the parameters of AR(2) process.

Figure 4.6-4.7 shows the result of this simulation which were obtained by averaging 100 independent simulations. We used ternary quantized input and desired signals for QLMS algorithm. As can be observed, the convergence of both algorithms are in excellent agreement.

4.5 Conclusion

In this chapter, we have described a novel adaptive filtering algorithm which makes use of coarse quantization of input and the desired signals to decrease the number of floating point multiplications in the required updates. Theoretically, we have shown that the QRLS algorithm running on the coarsely quantized signals solves exactly the same normal equations corresponding to the RLS case if the input signal is bounded in amplitude. Through error analysis and simulation studies, we have shown that very good performance was achieved. Although use of roughly quantized signals does not change the the order of the multiplications required by the conventional RLS algorithm, significant savings can be made in the fast RLS and LMS algorithms.

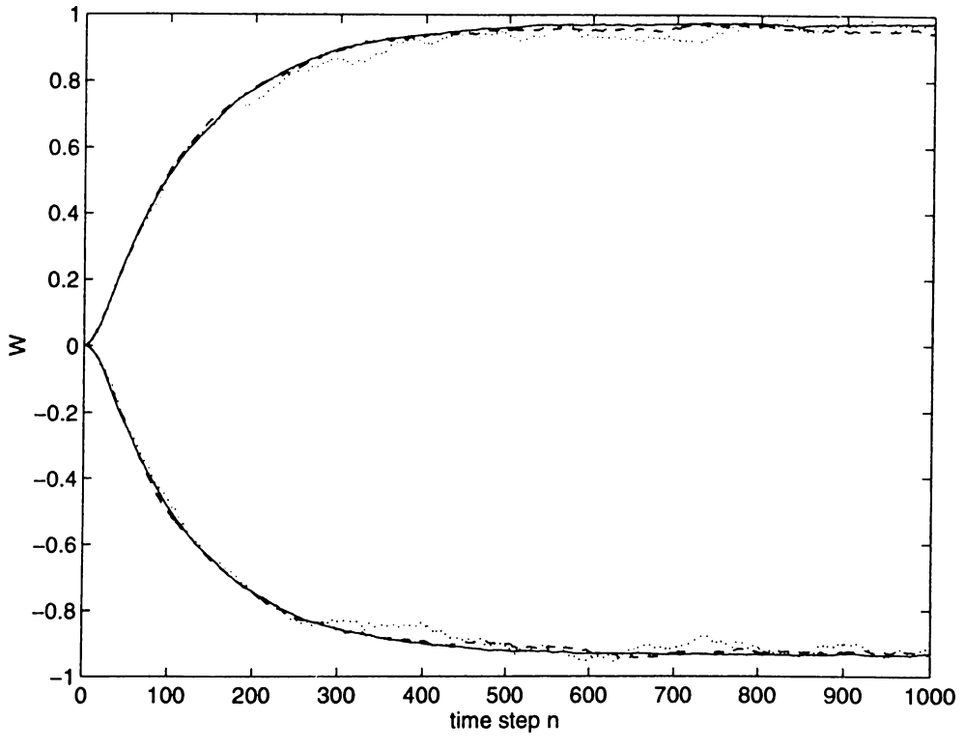

Figure 4.6: Convergence of LMS(solid), 3-bit QLMS(dash) and ternary QRLS(dot)

4.6 Appendix

In this appendix we will prove that random reference correlator, shown in Fig. 1, provides an unbiased estimate for the cross-correlation of two stationary sequences $x_1(n)$ and $x_2(n)$. In this appendix we will prove that random reference correlator, shown in Fig. 1, provides an unbiased estimate for the cross-correlation of two stationary sequences $x_1(n)$ and $x_2(n)$.

Consider two uniform midrise quantizers of step sizes Δ_1 and Δ_2 as in Fig. 3. Note that, the additive random reference signals $\eta_1(n)$ and $\eta_2(n)$ are zero mean white processes, which are independent of each other and of $x_1(n)$ and $x_2(n)$. Their distribution is uniform within one quantization level. Random reference signals $\eta_1(n)$ and $\eta_2(n)$ are added to the input signals to form two new processes

$$y_1(n) = x_1(n) + \eta_1(n) \tag{4.75}$$

$$y_2(n) = x_2(n) + \eta_2(n) \tag{4.76}$$

Resulting signals $y_1(n)$ and $y_2(n)$ are then quantized to obtain $y_{1q}(n)$ and

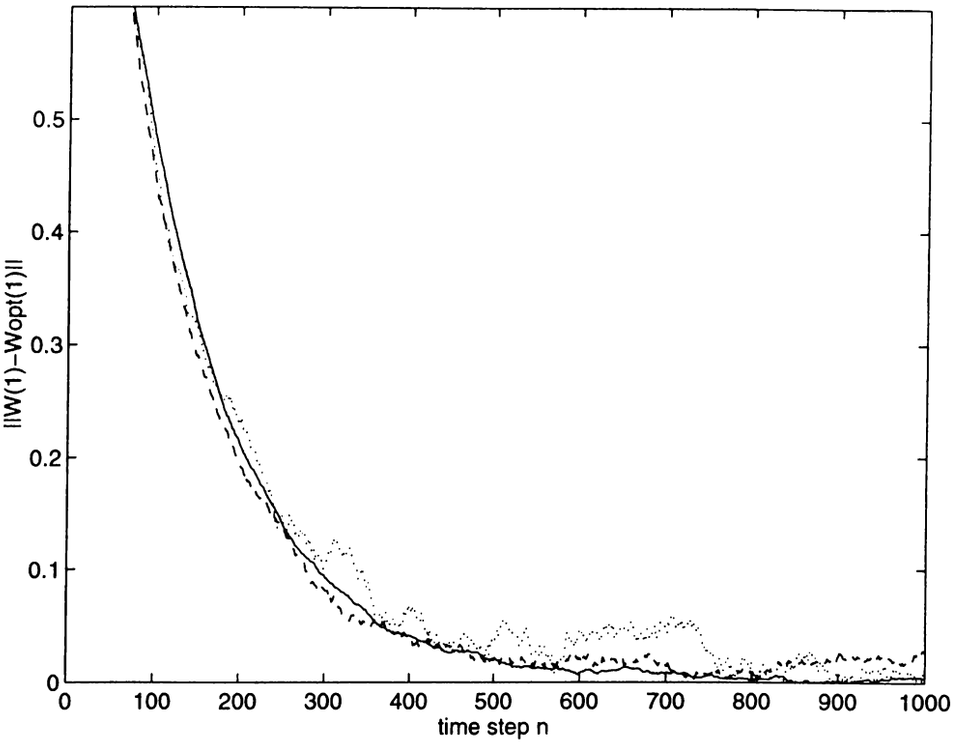

Figure 4.7: Coefficient deviations for LMS(solid), 3-bit QLMS(dash) and ternary QLMS(dot)

$y_{2q}(n)$. The cross-correlation function at lag m is

$$\begin{aligned} R_{y_{1q}y_{2q}}(m) &= E\left[y_{1q}(n)y_{2q}(n+m)\right] \\ &= E[Q_1(x_1(n)+\eta_1(n)) \times Q_2(x_2(n+m)+\eta_2(n+m))]. \end{aligned} \tag{4.77}$$

In the following lines, we will use x_1 and x_2 instead of $x_1(n)$ and $x_2(n+m)$ respectively and assume that $j = 1, 2$.

Let the value of input at time instant n be $x_j(n) = k_j\Delta_j + s_j$ with $0 < s_j < \Delta_j/2$. The requirement for positive s_j is only for convenience. The ensuing discussion can be extended to negative s_j also, in a very straightforward fashion. The range of stochastic reference signals is $(-\Delta_j/2,\ \Delta_j/2)$ as in Fig. 4.7. Two sub cases can be distinguished keeping in mind that $k_j\Delta_j < x_j < (k_j + 1/2)\Delta_j$

$$Q_j(x_j+\eta_j) = \begin{cases} (k_j+\frac{1}{2})\Delta_j & \text{with prob.} \quad \alpha_j = \frac{x_j-(k_j-\frac{1}{2})\Delta_j}{\Delta_j} \\ (k_j-\frac{1}{2})\Delta_j & \text{with prob.} \qquad 1-\alpha_j \end{cases}$$

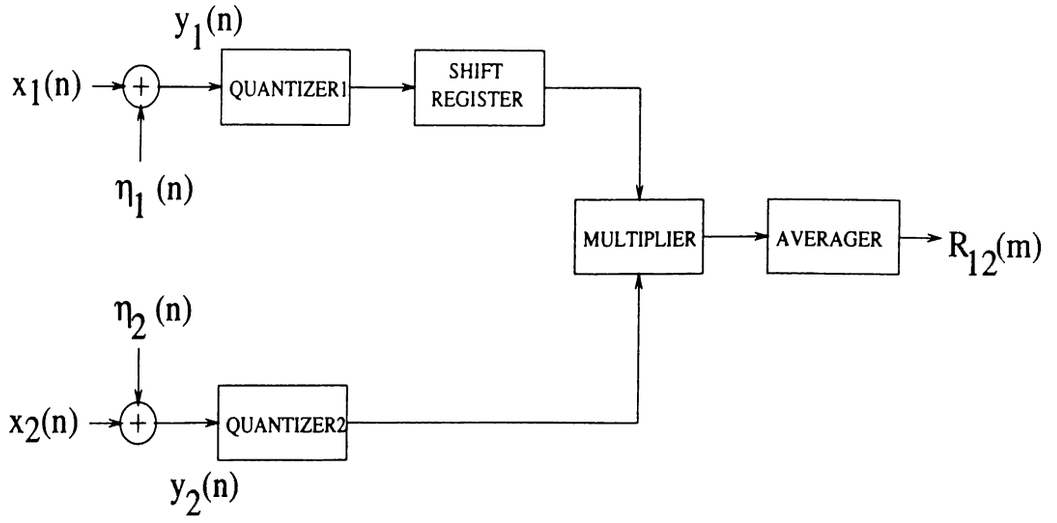

Figure 4.8: Random reference correlator.

Given the probabilities of x_j taking the values $(k \pm \frac{1}{2})\Delta_j$, we can take the expectation in (4.77) by conditioning on x_1 and x_2. With $I = E[y_{1q}y_{2q} \mid x_1\ ,\ x_2]$ it can be shown that

$$\begin{aligned} I &= \left(k_1+\frac{1}{2}\right)\Delta_1\alpha_1\left(k_2+\frac{1}{2}\right)\Delta_2\alpha_2 \\ &+ \left(k_1+\frac{1}{2}\right)\Delta_1\alpha_1\left(k_2-\frac{1}{2}\right)\Delta_2(1-\alpha_2) \\ &+ \left(k_1-\frac{1}{2}\right)\Delta_1(1-\alpha_1)\left(k_2+\frac{1}{2}\right)\Delta_2\alpha_2 \\ &+ \left(k_1-\frac{1}{2}\right)\Delta_1(1-\alpha_1)\left(k_2-\frac{1}{2}\right)\Delta_2(1-\alpha_2) \end{aligned} \tag{4.78}$$

Multiplying out the terms

$$\begin{aligned} I &= \Delta_1\Delta_2\left\{\left(k_1+\frac{1}{2}\right)\left(k_2+\frac{1}{2}\right)\left(\frac{x_1}{\Delta_1}-\left(k_1+\frac{1}{2}\right)\right)\left(\frac{x_2}{\Delta_2}-\left(k_2+\frac{1}{2}\right)\right)\right. \\ &+\left(k_1+\frac{1}{2}\right)\left(k_2-\frac{1}{2}\right)\left(\frac{x_1}{\Delta_1}-\left(k_1+\frac{1}{2}\right)\right)\left(\left(k_2+\frac{1}{2}\right)-\frac{x_2}{\Delta_2}\right) \\ &+\left(k_1-\frac{1}{2}\right)\left(k_2+\frac{1}{2}\right)\left(\left(k_1+\frac{1}{2}\right)-\frac{x_1}{\Delta_1}\right)\left(\frac{x_2}{\Delta_2}-\left(k_2+\frac{1}{2}\right)\right) \\ &\left.+\left(k_1-\frac{1}{2}\right)\left(k_2-\frac{1}{2}\right)\left(\left(k_1+\frac{1}{2}\right)-\frac{x_1}{\Delta_1}\right)\left(\left(k_2+\frac{1}{2}\right)-\frac{x_2}{\Delta_2}\right)\right\} \end{aligned}$$

After simplification, we get

$$I = \Delta_1\Delta_2\frac{x_1x_2}{\Delta_1\Delta_2}\left[\left(k_1+\frac{1}{2}\right)\left(k_2+\frac{1}{2}\right)-\left(k_1+\frac{1}{2}\right)\left(k_2-\frac{1}{2}\right)\right.$$

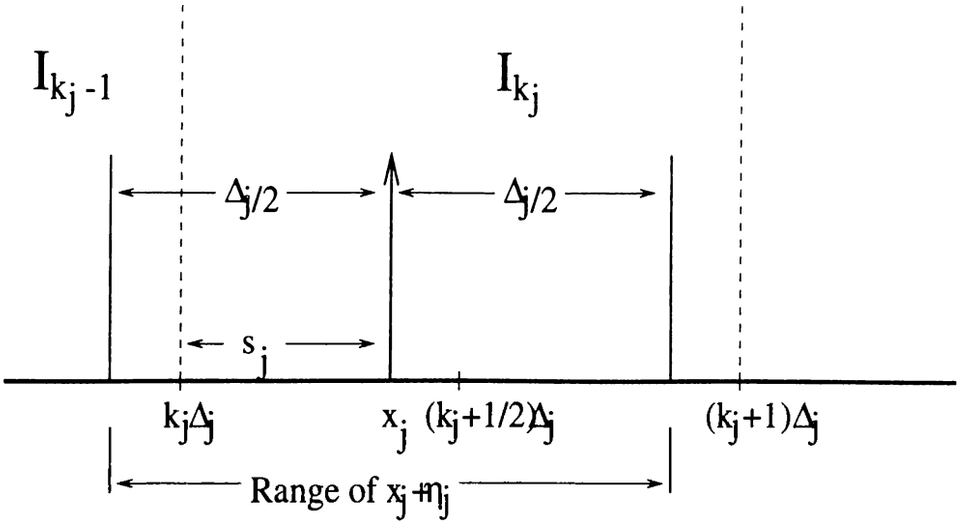

Figure 4.9: Uniform midrise quantizer. I_k and I_{k-1} are adjacent decision regions.

$$+\left(k_1-\frac{1}{2}\right)\left(k_2-\frac{1}{2}\right)-\left(k_1-\frac{1}{2}\right)\left(k_2+\frac{1}{2}\right)\Big]$$
$$= x_1x_2. \tag{4.79}$$

Hence, we obtain the desired equality

$$E[y_{1q}y_{2q}] = E\left[\,E[y_{1q}y_{2q}\mid x_1\,,\,x_2]\right] = E[x_1x_2]\ . \tag{4.80}$$

The result in (4.80) shows that the cross-correlation function of the process is the same before and after quantization provided that uniformly distributed random noise is added to the signals under consideration.

In showing the equivalence of cross-correlation function before and after quantization, a uniform midrise quantizer without a null zone is used. A general proof of this result for arbitrary quantizers can be found in [3].

Estimation error of the modified digital correlator has been studied in [3]. Mean square error, defined as the expected value of the square of the correlator output from the true value of the correlation function $R_{x_1x_2}(m)$ is given by

$$\epsilon(m;\ N) = \frac{1}{N}\left\{E[y_{1q}^2(0)y_{2q}^2(m) - E[x_1^2x_2^2]\right\} + \epsilon_c(m;\ N)$$

where $\epsilon_c(m;\ N)$ is the mean square error of the conventional direct correlator given by

$$\epsilon_c(m;\ N)=\frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}E[x_1(i)x_2(i+m)x_1(j)x_2(j+m)]-R_{x_1x_2}^2 \tag{4.81}$$

For a comparison, consider the mean square errors of correlators using various number of quantizations levels. Let $x_1(n)$ and $x_2(n)$ be the sample functions of statistically independent white Gaussian processes. The relative mean-square error $\epsilon_m(m;\ N)/\epsilon_c(m;\ N)$ for coarse quantization was found and tabulated in [3] and is also given in the following table for convenience

Levels	2	4	8	16	...	∞
$\frac{\epsilon_m(m;\ N)}{\epsilon_c(m;\ N)}$	72.23	2.75	1.23	1.054	...	1

As can be observed, a small increase in the number of quantization levels decreases the error sharply. For 8-level (3-bit) or 16-level (4-bit) quantizers, the mean-square error of the correlator is almost identical to that of the conventional direct correlator.

Chapter 5

CONCLUSION

As a result of the investigation presented in this thesis, novel adaptation algorithms are developed to improve the performances of the commonly used adaptation algorithms. The new algorithms can be classified into two general categories. The algorithms in the first category use filtered signals in the adaptation process. This way the condition number of the input autocorrelation matrix can be reduced with a result of faster convergence in the LMS adaptation. Also, by choosing the prefilter as a softlimiter, both the LMS and the RLS algorithms can be used in the in the presence of α-stable processes.

The second category consist of the algorithms that use roughly quantized signals in the adaptation process. Based on the analytical derivations, it has been shown that although significantly less number of multiplications are computed, the convergence point of these algorithms are the same as that of their conventional counterparts.

Based on the extensive set of simulations and analytical derivations, it can be concluded that the proposed algorithms are strong candidates to become the standard algorithms in their application areas.

REFERENCES

[1] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Prentice Hall, NJ, 1985.

[2] S. Haykin, *Adaptive Filter Theory*, Prentice Hall, NJ, 1986.

[3] C. F. N. Cowan and P. M. Grant, *Adaptive Filters*, Prentice Hall, NJ, 1985.

[4] S. Haykin, *Introduction to Adaptive Filters*, Macmillan Publishing Company, NY, 1984.

[5] M. G. Bellanger, *Adaptive Digital Filters and Signal Analysis*, Marcel Dekker Inc., NY, 1987.

[6] M. L. Honig and D. G. Messerschmitt, *Adaptive Filters*, Bell Communications Research Inc., 1984.

[7] P. M. Clarkson, *Optimal and Adaptive Signal Processing*, CRC Press, Florida, 1993.

[8] F. A. Graybill, *Matrices With Applications in Statistics*, Wadsworth, CA, 1983.

[9] P. J. Huber, *Robust Statistical Procedures*, Society For Industrial and Applied Math., PA, 1977.

[10] A. S. Householder, *The Theory of Matrices in Numerical Analysis*, Blaisdell, Waltham, 1964.

[11] N. Mohanty, *Signal processing*, Van Nostrand Reinhold Company, New York, 1987.

[12] Ke-Yen Chang and A. D. Moore, "Modified Digital Correlator and Its Estimation Errors", *IEEE Trans. on Information Theory* , vol. 16, no. 6, November 1970.

[13] H. Berndt, "Correlation Function Estimation by a Polarity Method Using Stochastic Reference Signals", *IEEE Trans. on Information Theory*, vol. 14, no. 6, pp. 796-801, November 1968.

[14] D. Landsberg and A. Cohen, "Fast Correlation Estimation by a Random Reference Correlator", *IEEE Trans. on Instrumentation and Measurement*, vol. IM-32, no. 3, pp. 438-442, September 1983.

[15] F. R. Lawson and C. D. McGillem, "The Use of High Sampling Rate and Ternary Quantization to Improve the Performance of the Random Reference Correlator", *IEEE Trans. on Information Theory*, pp. 269-271, March 1974.

[16] E. Masry, "The Application of Random Reference Sequences to the Reconstruction of Clipped Differentiable Signals", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, no. 6, pp. 953-963, December 1982.

[17] A. Cohen and D. Landsberg, "Adaptive Real-Time Wavelet Detection", *IEEE Trans. on Biomedical Engineering*, vol. BME-30, no. 6, pp. 332-340, June 1983.

[18] E. Masry, "The Reconstruction of Analog Signals From the Sign of Their Noisy Samples", *IEEE Trans. on Information Theory*, vol. IT-27, no. 6, pp. 735-745, November 1981.

[19] J. M. Cioffi and T. Kailath, "Fast, RLS, Transversal Filters for Adaptive Filtering," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 304-337, Apr. 1984.

[20] G. Carayannis, D. Manolakis, and N. Kalouptsidis, "A Fast Sequential Algorithm for Least Squares Filtering and Prediction," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 1392-1402, Dec. 1983.

[21] D. Falconer and D. Ljung, "Application of Fast Kalman Estimation to Adaptive Equalization", *IEEE Trans. Communications*, vol. COM-26, pp. 1439-1446, October 1978.

[22] J. M. Cioffi, " A Fast Adaptive ROTOR's RLS Algorithm", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, no. 4, pp. 631-653, April 1990.

[23] J. M. Cioffi, " An Unwindowed RLS Adaptive Lattice Algorithm", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, no. 3, pp. 365-371, March 1988.

[24] D. T. M. Slock and T. Kailath, " Fats Transversal Filters with Data Sequence Weighting", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, no. 3, pp. 346-358, March 1988.

[25] N. J. Bershad, "On Weight Update saturation Nonlinearities in LMS Adaptation,"*IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, no. 4, April 1988.

[26] S. C. Douglas and T. H. Y. Meng, "Stochastic Gradient Adaptation Under General Error Criteria", *IEEE Trans. on Signal Processing*, vol. 42, no. 6, pp. 1335-1345, June 1994.

[27] James R. Zeidler, "Performance Analysis of LMS Adaptive Filters," *Proceedings of IEEE*, vol. 78, no. 12, pp. 1781-1806, December 1990.

[28] O. Arikan, M. Belge, E. Cetin and E. Erzin, "Adaptive Filtering Approaches for Non-Gaussian Stable Processes," *ICASSP-95 Proceedings*, Detroit 1995.

[29] S.C. Douglas and T.H.Y. Meng, "The Optimum Scalar Data Nonlinearity in LMS Adaptation for Arbitrary i.i.d. Inputs," *IEEE Trans. on Signal Processing*, vol. 40, no. 6, pp. 1566-1570, June 1992.

[30] E. Eweda, " A Tight Upper Bound of the Average Absolute Error in a Constant Step-Size Algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, no. 11, pp. 1774-1780, November 1989.

[31] A. Nehorai and D. Malah, "On the Stability and Performance of the Adaptive Line Enhancer," *Proc. ICASSP 80*, vol.2, pp. 478-481, April 1980.

[32] E. Eweda and Odile Macchi, "Convergence of the RLS and LMS Adaptive Filters," *IEEE Trans. on Circuits and Systems*, vol. CAS-34, no. 7, July 1987.

[33] J. Jiang and R. Cook, "Fats Parameter Tracking RLS Algorithm With High Noise Immunity," *Electronics Letters*, vol. 28, no. 22, pp. 2043-2045, October 1992.

[34] J-L. Botto and G. V. Moustakides, "Stabilizing Fast Kalman Algorithms," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, no. 9, pp. 1342-1348, September 1989.

[35] Sumit Roy and J.J. Shynk, "Analysis of the Momentum LMS algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, no. 12, pp. 2088-2098, December 1988.

[36] B. Widrow and M.E. Hoff, Jr., " Adaptive Switching Circuits," Tech. Rep. 1553-1, Stanford Electron Labs., Stanford, CA, June 1960.

[37] S. S. Pillai, M. Harisankar, "Simulated performance of a DS spread spectrum system in impulsive atmospheric noise," *IEEE Trans. Electromagnetic Compat.*, vol. 29, pp. 80-82, 1987.

[38] M. Bouvet and S. C. Schwartz,"Comparison of adaptive and robust receivers for signal detection in ambient underwater noise," *IEEE Trans. Acoust. Speech and Signal Proc.*, vol. 37, pp. 621-626, 1989.

[39] Geoffrey A. Williamson, Peter M. Clarkson and William A. Sethares, "Performance Characteristic of the Median LMS adaptive filter," *IEEE Trans. on Signal Processing*, vol. 41, pp. 667-680, 1993

[40] M. Shao and C. L. Nikias, "Signal Processing with fractional lower order moments: Stable Processes and their applications," *Proc. IEEE*, vol. 81, pp. 986-1009, 1993.

[41] S. A. Kassam and H. V. Poor, "Robust techniques for signal processing: a survey," *Proc. of IEEE*, vol.73, pp.433-481, March 1985.

[42] D. F. Marshall, W. K. Jenkins and J. J. Murphy, "The use of orthogonal transforms for improving performance of adaptive filters," *IEEE Transactions on Circuits and Systems*, vol. 36, pp. 474-484, 1989.

[43] F. Beaufays, "Transform-domain adaptive filters: an analytical approach,", *IEEE Transactions on Signal Processing*, vol. 43, pp. 422-431, 1995.

[44] S. Florian and N. J. Bershad, "A weighted normalized frequency domain LMS adaptive algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, pp. 1002-1007, 1988